Large-Scale Training of Generative Adversarial Networks

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät

der Eberhard Karls Universität Tübingen

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

(Dr. rer. nat.)

vorgelegt von

Axel Sauer

aus Bad Mergentheim

Tübingen

2023

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	10.10.2024
Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter/-in:	Prof. Dr. Andreas Geiger
2. Berichterstatter/-in:	Prof. Dr. Matthias Bethge

Abstract

Generative models have transformed various fields, particularly image synthesis, by providing a data-centric approach to visual content creation. This thesis investigates generative adversarial networks (GANs), a prominent class of generative models recognized for their fast inference and control over synthesized outputs. Despite their advantages, GANs encounter difficulties when scaling to large, diverse datasets, such as training instability and mode collapse. To tackle these issues, we devise novel techniques and network architectures for scaling GANs. Our contributions encompass Projected GANs, which achieve state-of-the-art performance on 22 benchmark datasets and train up to 40 times faster; StyleGAN-XL, a GAN trained on ImageNet, constituting the new state-of-the-art; and StyleGAN-T, a large-scale text-to-image synthesis model, the first GAN surpassing one billion parameters in model-size. By progressing towards developing the first GAN foundation model, we aspire to further enhance the field of generative artificial intelligence, expanding its impact on image synthesis and unlocking new applications, such as real-time image editing, personalized content generation, and interactive virtual experiences.

Kurzfassung

Generative Modelle haben verschiedenste Bereiche transformiert, insbesondere die Bildsynthese, durch einen datenzentrierten Ansatz zur Erstellung visueller Inhalte. Diese Arbeit untersucht Generative Adversarial Networks (GANs), eine Klasse von generativen Modellen, die für ihre schnelle Inferenz und Kontrolle über synthetisierte Ergebnisse bekannt sind. Trotz ihrer Vorteile stoßen GANs auf Schwierigkeiten beim Skalieren auf größere und vielfältigere Datensätze, wie zum Beispiel Trainingsinstabilität und Modus-Kollaps. Um diese Probleme zu lösen, entwickeln wir neue Techniken und Netzwerkarchitekturen zur Skalierung von GANs auf große Datensätze. Unsere Beiträge umfassen Projected GANs, die den neuen Stand der Technik auf 22 Benchmark-Datensätzen darstellen und bis zu 40-mal schneller trainieren; StyleGAN-XL, ein auf ImageNet trainiertes Modell, das hochauflösende Bilder (1024x1024) generiert; und StyleGAN-T, ein Text-zu-Bild-Synthesemodell, das erste GAN, das aus mehr as einer Milliarde Parameter besteht. Indem wir uns auf die Entwicklung des ersten GAN-Foundation Modell zubewegen, streben wir danach, das Feld der generativen künstlichen Intelligenz weiter zu verbessern, seinen Einfluss auf die Bildsynthese zu erweitern und neue Anwendungen freizuschalten, wie zum Beispiel Echtzeit-Bildbearbeitung, personalisierte Inhaltsgenerierung und interaktive virtuelle Erlebnisse.

Acknowledgments

I'm genuinely grateful to my supervisor, Andreas Geiger, for giving me the chance to join his high-profile research group. This journey kicked off with his mentorship during my master's thesis, a defining moment that set me on the path toward a career in computer science. After my initial attempt at a Ph.D. didn't meet my expectations, he offered me a second chance, a decision that has positively shaped and greatly impacted my life. Andreas has been a consistent and supportive guide throughout my research, sharing his extensive knowledge and inspiring me with his dedication to high-quality work. He entrusted me with the flexibility to choose my research topics and determine the level and timing of his involvement, fostering a learning experience tailored to my needs.

I am grateful to my university co-authors, Kashyap Chitta, Jens Müller, Katja Schwarz, Michael Niemeyer, and Yiyi Liao. Our collaboration on various projects has been both highly enjoyable and an insightful learning experience. I truly appreciated our many discussions about research, and the shared experience of meeting tight deadlines was particularly memorable.

Furthermore, my gratitude goes to my co-authors at NVIDIA, Tero Karras, Samuli Laine, and Timo Aila, for a fantastic internship experience. Working alongside you has been a true privilege and undeniably one of the standout experiences during my Ph.D. journey.

Additionally, I'd like to express my gratitude to all the colleagues I've met at the University of Tübingen and NVIDIA. It's been a true pleasure interacting with each one of you. Some of these professional relationships have grown into personal friendships, which I consider myself very fortunate to have.

My heartfelt appreciation extends to my family, friends, and especially my wife, Vanessa, whose presence in my life I can't appreciate enough. Vanessa's support has been invaluable throughout my Ph.D., in moments of triumph and during the most challenging times. It's a privilege to share my dreams, thoughts, and life journey with you.

Contents

Abstract **iii**

Kurzfassung **v**

Acknowledgments **vii**

1 Introduction **1**
1.1 Motivation 1
1.2 Applications of Generative Artificial Intelligence 2
1.3 Contributions of the Thesis 3
1.4 Outline of the Thesis 3

2 Background **5**
2.1 Deep Generative Modeling 5
2.1.1 Autoregressive Models 6
2.1.2 Diffusion Models 7
2.1.3 Generative Adversarial Networks 8
2.2 The Basics of GAN Training 9
2.3 StyleGAN: A Style-Based Generator Architecture 12
2.3.1 Network Architecture 12
2.3.2 The Evolution of StyleGAN 13
2.3.3 Advanced Training Methods 14
2.4 Scaling Neural Networks 15
2.4.1 Scaling Model Size 15
2.4.2 Scaling Compute 16
2.4.3 Scaling Datasets 17

3 Leveraging Pretrained Feature Networks for GAN Training **19**
3.1 Introduction 19
3.2 Projected GANs 20
3.2.1 Definition 20
3.2.2 Consistency 21
3.2.3 Multi-Scale Discriminators 23
3.2.4 Random Projections 23
3.2.5 Pretrained Feature Networks 24
3.3 Experiments 25
3.3.1 Ablation Study 25

3.3.2 Quantitative Comparison to State-of-the-Art 28
3.4 Discussion 36

4 Large-Scale Class-Conditional Image-Synthesis with GANs 37
4.1 Introduction 37
4.2 StyleGAN-XL 38
4.2.1 Adapting Regularization and Architectures 39
4.2.2 Reintroducing Progressive Growing 41
4.2.3 Exploiting Multiple Feature Networks 43
4.2.4 Classifier Guidance for GANs 43
4.3 Experiments 44
4.3.1 Image Synthesis 45
4.3.2 Inversion and Manipulation 45
4.4 Discussion 55

5 Large-Scale Text-to-Image Synthesis with GANs 59
5.1 Introduction 59
5.2 StyleGAN-T 60
5.2.1 Redesigning the Generator 62
5.2.2 Redesigning the Discriminator 64
5.2.3 Variation vs. Text Alignment Tradeoffs 65
5.3 Experiments 67
5.3.1 Configuration Details 67
5.3.2 Quantitative Comparison to State-of-the-Art 68
5.3.3 Evaluating Variation vs. Text Alignment 69
5.3.4 Qualitative Results 69
5.4 Discussion 70

6 Quantifying Progress: Projected GAN to StyleGAN-XL to StyleGAN-T 77
6.1 Comparing Configurations 77
6.2 Comparing Performance 78
6.3 Discussion 79

7 Conclusion 81
7.1 Limitations and Future Work 81
7.2 The Future of GANs in the Generative Model Landscape 82

A Credits 85

Bibliography 87

List of Figures

2.1 **1D example.** GANs are trained by simultaneously updating the discriminative distribution, $D(\mathbf{x})$, to discriminate between samples from the data generating distribution, p_{data}, and samples from the generative distribution, p_{model}. The lower horizontal line indicates the domain from which $\mathbf{z}$ is sampled, in this case, from a standard normal distribution. The horizontal line above represents part of the domain of x. The upward arrows illustrate how the mapping $\mathbf{x} = G(\mathbf{z})$ transforms the initial z distribution. **(a)** At the start of training, p_{data} and p_{model} have large non-overlapping regions. **(b)** After an update to G, the gradient of D has guided G(z) to flow to regions more likely to be classified as data. **(c)** After several steps of training, G and D reach a point where both cannot improve further as p_{data} equals p_{model}. At this stage, the discriminator cannot differentiate between the two distributions, resulting in $D(x) \approx 0.5$, indicating that D is equally likely to classify a sample as coming from either the data distribution or the model distribution. The example and figures are from [74]. 11

2.2 **Comparative overview of the generator architecture evolution in the StyleGAN series.** (a) A standard GAN uses a single, direct mapping from the latent code $\mathbf{z}$ to the image. (b) StyleGAN introduces a two-step process with a mapping network $\mathbf{G}_m$ and a synthesis network $\mathbf{G}_s$. It first maps the input to an intermediate latent space W, controlling the generator through Adaptive Instance Normalization (AdaIN) at each convolution layer. Noise B is added after each convolution, before the nonlinearity. The mapping network consists of 8 layers and the synthesis network has 18 layers — two for each resolution from 4×4 to 1024×1024. (c) StyleGAN2 reduces artifacts by shifting normalization from individual feature maps to convolutional kernels. It introduces changes to the original architecture, including removing redundant operations at the beginning, moving the addition of bias b_i and noise B to be outside the active area of a style, and adjusting only the standard deviation per feature map. This revised architecture allows for the replacement of AdaIN with a *demodulation* operation applied to the weights of each convolution layer. (d) StyleGAN3 addresses the *texture sticking* problem using alias-free operations. The main path of this generator consists of Fourier features and normalization, modulated convolutions, and filtered nonlinearities. The figure is compiled of illustrations from [115, 116, 114]. 13

2.3 **Artifacts in StyleGAN and StyleGAN2.** Left: Adaptive Instance Normalization (AdaIN) in StyleGAN leads to systemic water droplet-like artifacts in all generated images, evident in all feature maps from the 64x64 resolution onward. Right: An example of *texture sticking*. StyleGAN2 retains hair details at the same coordinates during a latent space interpolation, resulting in unwanted horizontal streaks. Ideally, the hair should animate smoothly, creating a time-varying field as demonstrated by the alias-free StyleGAN3. The visualizations are from [115, 116]. 14

2.4 **Evolving Complexity in Generative Modeling Datasets.** Shown are sample images from a selection of datasets used in generative modeling research. From left to right: CIFAR10 (2009) [125], FFHQ (2020) [116], ImageNet (2009) [48], and LAION-5B (2022) [208]. These samples underscore the increasing complexity trend in datasets over time. Despite FFHQ being introduced later than ImageNet, it's noteworthy that generative models producing sample quality on par with those trained on FFHQ only appeared recently [53, 92]. 17

3.1 **CCM and CSM.** Left: CCM (dashed blue arrows) employs 1×1 convolutions with random weights. Right: CSM (dashed red arrows) adds random 3×3 convolutions and bilinear upsampling, yielding a U-Network. 24

3.2 **Training Properties.** Left: Projected FastGAN surpasses the best FID of StyleGAN2 (at 88 M images) after just 1.1 M images on LSUN-Church. Right: Projected FastGAN yields significantly improved FID scores, even when using subsets of CLEVR with 1k and 10k samples. 29

3.3 **Training progress on LSUN church at 256^2 pixels.** Shown are samples for a fixed noise vector $\mathbf{z}$ over k images. From top to bottom: FastGAN, StyleGAN2-ADA, Projected GAN. 29

3.4 **Signed Discriminator Logits.** For this experiment, we project through F and train with up to four discriminators; we leave the augmentation probability constant. ($|D_i| = 1$: red, $|D_i| = 2$: blue, $|D_i| = 3$: pink, $|D_i| = 4$: green, RGB baseline: orange). For projected GAN training, the logits remain stable throughout training. 30

3.5 **Real samples (top rows) vs. samples by Projected GAN (bottom rows).** Datasets (top left to bottom right): CLEVR (256^2), LSUN church (256^2), Art Painting (256^2), Landscapes (256^2), AFHQ-wild (512^2), Pokemon (256^2), AFHQ-dog (512^2), AFHQ-cat (512^2). 32

3.6 **Limitations.** Left: "Floating Heads." Right: Artifacts on FFHQ. 36

4.1 Class-conditional samples generated by StyleGAN3 (left) and StyleGAN-XL (right) trained on ImageNet at resolution 256^2. 38

4.2 **Training StyleGAN-XL**. We feed a latent code **z** and class label **c** to the pretrained embedding and the mapping network $\mathbf{G}_m$ to generate style codes **w**. The codes modulate the convolutions of the synthesis network $\mathbf{G}_s$. During training, we gradually add layers to double the output resolution for each stage of the progressive growing schedule. We only train the latest layers while keeping the others fixed. $\mathbf{G}_m$ is only trained for the initial 16^2 stage and remains fixed for the higher-resolution stages. The synthesized image is upsampled when smaller than 224^2 and passed through a CNN and a ViT and respective feature mixing blocks (CCM+CSM). At higher resolutions, the CNN receives the unaltered image while the ViT receives a downsampled input to keep memory requirements low but still utilize its global feedback. Finally, we apply eight independent discriminators on the resulting multi-scale feature maps. The image is also fed to classifier CLF for classifier guidance. 39

4.3 **Flexible Layer Specification of Stylegan-XL**. StyleGAN-XL consists of 39 layers at resolution 1024^2. Cutoff (blue) and minimum acceptable stopband frequency (orange) obey geometric progression over the layers; sampling rate (red) and actual stopband (green) are computed according to our design constraints. 42

4.4 **Samples at Different Resolutions Using the Same w**. The samples are generated by the models obtained during progressive growing. We upsample all images to 1024^2 using nearest-neighbor interpolation for visualization purposes. Zooming in is recommended. 47

4.5 **Interpolations.** StyleGAN-XL generates smooth interpolations between samples of different classes. 47

4.6 **Samples on FFHQ** 1024^2. 48

4.7 **Samples on Pokemon** 1024^2. 49

4.8 **Qualitiative Comparison on ImageNet** 256^2**.**. We compare BigGAN (left column), ADM (middle column), and StyleGAN-XL (right column). Classes from top to bottom: pizza, valley, daisy, dough, comic book. . . . 50

4.9 **Qualitiative Comparison on ImageNet** 256^2**.**. We compare BigGAN (left column), ADM (middle column), and StyleGAN-XL (right column). Classes from top to bottom: bulbul, nematode, jack-o'-lantern, balloon, crossword puzzle. 51

4.10 **Qualitiative Comparison on ImageNet** 256^2**.**. We compare BigGAN (left column), ADM (middle column), and StyleGAN-XL (right column). Classes from top to bottom: agaric, orange, Tibetian mastiff, espresso, paddlewheel. 52

4.11 **Inversion of a Given Source Image**. For BigGAN, we invert to its latent space **z**, for StyleGAN-XL we invert to style codes **w**. 54

4.12 **Interpolations**. StyleGAN-XL generates smooth interpolations between samples of different classes (Row 1 & Row 2). PTI allows inverting to the latent space with low distortion (outermost image, Row 3 & Row 4), and consistently embeds out-of-domain inputs, such as the one on the bottom right. 54
4.13 **Image Editing and Style Mixing**. Left: First, a given image is inverted via PTI [190]. Right: Given two images, we can mix their styles. This methods works for samples of the same or similar classes, and to a certain extent, for distant classes. For this experiment, we utilize random samples instead of inversions. 56
4.14 **Image Manipulation via Language**. Given a random sample, we manipulate the image by by following semantic directions in latent space found by StyleMC [120]. The latent space directions from top to bottom are: "smile", "no stripes", and "big eyes". 56
4.15 **Imagenet Classes Containing Humans**. Samples for BigGAN and ADM are taken from [53]. 57

5.1 **Quality vs. speed** in large-scale text-to-image synthesis. StyleGAN-T greatly narrows the quality gap between GANs and other model families while generating samples at a rate of 10 FPS on an NVIDIA A100. The y-axis corresponds to zero-shot FID on MS COCO at 256×256 resolution; lower is better. 60
5.2 **Example images and interpolations.** StyleGAN-T generates diverse samples matching the text prompt and allows for smooth interpolations between prompts, illustrated as a single continuous interpolation in scanline order. Generating these 56 samples at 512×512 takes 6 seconds on an NVIDIA RTX 3090, while a comparable grid takes up to several minutes with current diffusion models. 61
5.3 **Overview of StyleGAN-T. (a)** Our generator architecture (Section 5.2.1) is closely related to StyleGAN2, with the learned constant replaced with Fourier features and conditioning applied in a slightly different place. **(b)** For each resolution, a generator block is executed and its contribution is accumulated to the image via a dedicated ToRGB layer. The generator blocks employ residual connections and a new 2^{nd} order style mechanism (Eq. 5.1). **(c)** Our discriminator (Section 5.2.2) processes the intermediate tokens of a DINO-trained vision transformer using 5 identical discriminator heads. Text conditioning is done using projection at the end. **(d)** Text prompt is embedded using CLIP and supplied to the generator and discriminator. We also employ a guidance term to further improve text alignment (Section 5.2.3). 62
5.4 **Truncation.** Four samples for the prompt "a graphite sketch of Eva Longoria" with different random $\mathbf{z}$. Increasing truncation (decreasing ψ) improves the text alignment according to mean CLIP score per row ($\overline{CS}$) at the cost of lower variation. 66

5.5 **Comparing text alignment tradeoffs.** We compare FID–CLIP score curves of StyleGAN-T, distilled Stable Diffusion (SD-distilled), and eDiff-I. We report values of SD-distilled at a guidance scale of $w = 4$. For a fair comparison, we report numbers for CLIP-conditioned eDiff-I disabling additional conditioning on T5-XXL text embeddings. The models use different methods to increase the CLIP score (i.e., text alignment): StyleGAN-T decreases truncation $\psi = \{1.0 \dots 0.0\}$, SD-distilled increases the number of sampling steps $\{2, 4, 8\}$, eDiff-I increases guidance scale $w = \{0 \dots 10\}$. . 70
5.6 **Text encoder training.** Training the CLIP text encoder (TE) pushes the entire FID–CLIP score curve to the right, hence, increasing overall text alignment. 71
5.7 **Latent manipulation.** Samples (first column) can be manipulated by following semantic directions in latent space. 72
5.8 **Styles.** Samples generated by StyleGAN-T for a fixed random seed and the caption "astronaut, {X}", where X is denoted below each image. . . . 73
5.9 **Additional truncation grids.** We show samples for 6 different prompts and 5 different random latents, shared between the prompts. Increasing truncation (decreasing ψ), improves the text alignment according to mean CLIP score per row, $\overline{CS}$, at the cost of lower variation. 74
5.10 **Qualitative Comparisons.** We show samples for 6 different prompts and 5 different random latents, shared between the prompts. For StyleGAN-T, we set $\psi = 0.6$. LDM and Stable Diffusion utilize 250 and 50 sampling steps, respectively, utilizing the DDIM / PLMS sampler [143]. For DALL·E 2, we generate images via the official DALL·E service [168]. . . 75
5.11 **Failure cases.** StyleGAN-T can fail to bind attributes to objects or generate separate entities (top row) and to produce coherent text (middle row). Furthermore, the model struggles at high-resolution, resulting in samples with low details and text coherence (bottom row). 76

6.1 **Uncurated Results for FFHQ (256^2).** For a fair comparison, we show untruncated samples. The images are selected randomly given one global random seed. We recommend zooming in for comparison. 80

List of Tables

3.1 **Feature Space Fréchet Distances.** We aim to find the best combination of discriminators and random projections to fit the distributions in feature network F. We show the relative FD at different layers of F ($rel\text{-}FD_i$) between 50k generated and real images on LSUN-Church. $rel\text{-}FD_i$ is **normalized** using the baseline Fréchet Distances for a model with a standard single RGB image discriminator. Hence, values > 1 indicate worse performance than the RGB baseline. We report *rel-FD* for four layers of an EfficientNet (L_1, L_2, L_3 and L_4 from shallow to deep), as well as relative Fréchet Inception Distance (FID) [90]. Note that $rel\text{-}FD_i$ should not be compared between different feature spaces, i.e., only within-column comparisons are meaningful. Blue boxes highlight the layers which we supervise via independent discriminators. The green box corresponds to a perceptual discriminator [223], which takes in all feature maps at once. . 26

3.2 **Pretrained Feature Networks Study**. We train the projected GAN with different pretrained feature networks. We find that compact EfficientNets outperform both ResNets and Transformers. 27

3.3 **Quantitative Results.** Projected GAN* reports the point where our approach surpasses the state-of-the-art. StyleGAN2* obtains the lowest FID in previous literature if trained long enough. 33

3.4 **Metrics on Large Datasets** (256^2)**.** Projected GAN compares favorably on most metrics. Exceptions are precision on FFHQ, Cityscapes, and LSUN Church. As argued by [116], shifting from precision to recall is generally desirable, since recall can be traded into precision via truncation. 34

3.5 **Metrics on Small Datasets** (256^2)**.** Projected GAN performs best on most metrics. 35

4.1 **Ablation Study on ImageNet** 128^2. Results for different configurations after training for 15 V100-days. 39

4.2 **Feature Network Study on ImageNet** 128^2. Comparing combinations of different feature networks **F**. Beginning from the base configuration using an EfficientNet-lite0 (EffNet), we add a second **F** with varying architecture type and pretraining objective (*Class*: Classification, *Self*: MoCo-v2 [38]). 44

4.3 **Image Synthesis on ImageNet**. Empty cells indicate that the model was not available and the respective metric was not evaluated in the original work. 46

4.4 **Inversion Results.** The metrics are computed between the inversions obtained by the model and the reconstruction targets. 53

4.5 **Inference speed comparison.**. We measure the time required for a forward pass with batch size 1 in V100-seconds. ADM uses classifier guidance. . 57

5.1 **Architecture ablation.** Our architectural changes notably improve sample quality and text alignment. Here, we use the lightweight training configuration described in Section 5.3.1. 63
5.2 **Generator, discriminator, and training hyperparameters** for the two setups used in this paper: Lightweight and Full configuration. 67
5.3 **Training schedules** for the two training configurations used in this paper. The times are listed as the number of days it would have taken on a single NVIDIA A100 GPU. An iteration corresponds to 2048 real and generated examples. 68
5.4 **Comparison of FID on MS COCO 64×64.** Inference speeds are measured on an A100. For LAFITE we estimate what its speed would be at a native 64×64 resolution. 68
5.5 **Comparison of FID on MS COCO 256×256.** Inference speeds are measured on an A100, except for Imagen and Parti that use a faster TPUv4 accelerator. The Stable Diffusion numbers are from [16, 130]; the other numbers are obtained from the respective papers or through correspondence with the authors. 69

6.1 **Generator, discriminator, guidance, and training hyperparameters** for the introduced approaches. We compare generators for output resolution 512x512 pixels. 79

1 Introduction

1.1 Motivation

The conventional approach to image synthesis, which involves digitally creating and manipulating visual content, has been primarily grounded in computer graphics techniques [69]. These techniques focus on generating photorealistic images with direct control over semantic attributes, typically achieved by creating meticulously designed 3D models rendered using realistic camera and illumination models. In contrast, Generative AI (GenAI) revolutionized various fields, particularly visual content creation, by offering a data-centric approach. GenAI incorporates a range of generative models, such as generative adversarial networks (GANs) [79] and diffusion models (DMs) [220], producing diverse outputs based on the dataset they were trained on.

Contrary to traditional computer graphics methods, GenAI emphasizes the design of training procedures, datasets, and input prompts instead of individual assets. This shift brings several advantages: faster iteration and refinement of ideas and concepts, improved results in challenging settings, such as movie dubbing [20] and photo filters [105], democratization via open-source and user-friendly tools [68, 80], and cost-effectiveness through reduced manual labor.

Recently, GenAI has experienced a shift towards the utilization of *foundation models* [21]. Foundation models are pretrained on large-scale datasets and can be adapted for different tasks with minimal finetuning. By transferring knowledge from their large-scale pretraining stage, these models generalize more effectively and outperform task-specific models. As a result, foundation models for image synthesis [183, 191, 194] have led to new applications, such as synthesizing a given subject in different scenes and styles [73, 193], text-based image and video editing [65, 157, 26], and text-to-3d synthesis [176, 138].

Today's foundation models for visual data are primarily DMs. Well-suited for scale training due to their stability and ability to manage large multi-modal datasets, DMs model the data generation process as a reverse diffusion process, starting from a simple noise distribution and gradually denoising to obtain the final data sample.

GANs are another family of generative models, well known for their inference speed and control over synthesized results via latent space manipulations. GANs consist of two neural networks, a generator and a discriminator, that are trained simultaneously in a competitive manner. The generator creates synthetic data, while the discriminator evaluates the generated data's authenticity. The objective is to improve the generator's performance until the discriminator can no longer differentiate between real and generated data. GANs require only a single forward pass and are thus much faster than DMs. There has been notable progress in speeding up DMs [199, 112, 146], yet, GANs remain significantly faster. High inference speed is crucial as it enables real-time applications and drastically reduces

operating costs, which is especially valuable for emerging AI-as-a-service platforms such as Adobe Firefly [4] and OpenAI DALL-E [169]. However, GANs have faced challenges when scaling to larger and more diverse datasets, such as training instability and mode collapse (where the model generates limited variations). Despite their unique advantages, GANs have yet to reach their full potential in large-scale applications.

This thesis addresses these challenges by developing techniques for scaling GANs to leverage their strengths. We will explore the challenges of scaling GANs to larger and more diverse datasets and propose novel approaches to overcome these challenges. Ultimately, our goal is to take steps toward building the first GAN foundation model, which combines the benefits of foundation models with the strengths of GANs. By achieving this, we aim to further advance the field of GenAI and its impact on image synthesis, opening up new possibilities for applications such as real-time image editing, personalized content generation, and interactive virtual experiences.

1.2 Applications of Generative Artificial Intelligence

GenAI has numerous applications, including image synthesis [17], natural language processing [27], drug discovery [23], code generation [36], and mathematics [133]. This thesis primarily focuses on improving GANs for image synthesis, and we will highlight potential applications for image synthesis models in the following.

Entertainment and Media. GenAI can streamline the creation of visual effects, animations, or entire scenes in movies or television shows, significantly reducing time and effort for manual content creation. GenAI also has potential applications in gaming, enhancing the realism and diversity of game environments.

Design and Fashion. GenAI has the potential to generate original artwork, graphic designs, or illustrations by learning from existing art styles and datasets. Artists and designers can use these models as creative tools to explore new styles, generate ideas, or enhance their creations. In the fashion industry, generative models can facilitate the design of new clothing, accessories, or patterns by learning from prevailing trends and styles. This approach can support designers in discovering innovative concepts and predicting future trends.

Advertising and Marketing. GenAI can generate personalized advertisements, product mockups, or promotional materials tailored to specific target audiences or preferences, enabling businesses to create more engaging and effective marketing campaigns.

Synthetic Training Data. GenAI can create synthetic training data for applications where real data is scarce or expensive, effectively addressing data limitations in various domains. Training models on a mix of synthetic and real data makes it possible to achieve more robust and accurate models that can handle diverse situations [201, 14]. Additionally, GenAI provides precise control over the generated data, thereby improving their suitability for various applications, such as dense visual alignment [172].

1.3 Contributions of the Thesis

In this thesis, we propose several novel training strategies and neural networks architecture for GANs:

- Projected GANs, which project real and generated samples into a pretrained feature space and mix features across channels and resolutions. This approach achieves state-of-the-art performance on 22 benchmark datasets, up to 40 times faster to train than previous methods.
- StyleGAN-XL, a generative model trained on ImageNet [48] using Projected GAN's powerful neural network priors and a tailored, progressive growing strategy. StyleGAN-XL generates high-quality images at a resolution of 1024×1024 and can invert and edit images of diverse object classes.
- StyleGAN-T, a large-scale text-to-image synthesis model that addresses the requirements of stable training on large image-text datasets, controllable fidelity vs. text alignment tradeoff, and strong text alignment. StyleGAN-T achieves state-of-the-art performance at a resolution of 64×64 and outperforms previous GANs at higher resolutions.

The above-mentioned contributions are part of three research papers [203, 206, 205] published at machine learning and computer graphics conferences. Please see Appendix A for a credits discussion.

1.4 Outline of the Thesis

We have organized the thesis into six sections. In Chapter 2, we provide an overview of generative modeling and delve into the landscape of models, with a particular emphasis on examining StyleGAN. We also review the challenges of distributed training of neural networks and discuss how training data is obtained at scale. Next, we introduce a new paradigm for training GANs in Chapter 3, which we refer to as *Projected GANs*. Chapter 4 details our training strategy for scaling GANs for image synthesis on ImageNet. In Chapter 5, we develop a method for scaling GANs to hundreds of millions of image-text pairs. Chapter 6 contrasts the three different approaches in a comparative study. Finally, Chapter 7 concludes the thesis by discussing the general limitations of GANs and assessing their potential future role in the generative model landscape.

2 Background

Our primary aim is to explore the scalability of GANs. To lay the foundation for our investigation, we provide an overview of the generative model landscape, elucidating the differences between GANs and other models. We begin by discussing the core principles of generative modeling. Next, we offer a comparative analysis of three key generative model classes: Autoregressive Models (ARMs), which decompose sequences into conditional probabilities; Diffusion Models (DMs), employing a noise-removing iterative process; and GANs, built on a competitive framework between a generator and a discriminator. Building on this general overview, we delve deeper into the basics of GAN training, followed by a discussion of StyleGAN, a modern instantiation of GANs. Lastly, we discuss common strategies for scaling neural network training and datasets.

2.1 Deep Generative Modeling

Generative modeling is a branch of machine learning that aims to learn the underlying probability distribution $p_{\text{data}}(\mathbf{x})$ of the data $\mathbf{x}$, enabling the creation of new data samples that resemble the training data. These models capture the inherent structure and patterns within the data. Deep generative modeling, in particular, utilizes deep neural networks [78], which are flexible and powerful, making them a popular choice for parameterizing generative models. Moving forward, our discussion will focus exclusively on deep generative models. Generative models can be categorized into four main groups following Tomczak [228]:

- **Autoregressive Generative Models (ARM):** ARMs generate new data samples by decomposing the joint probability distribution of a sequence into a product of conditional probabilities. This approach allows the model to generate data sequentially, one element at a time.

- **Flow-Based Models:** Flow-based models utilize invertible transformations to map data from a simple distribution (e.g., Gaussian) to a more complex one. This bijective mapping enables efficient sampling and exact likelihood computation.

- **Latent Variable Models:** These models learn a compact, lower-dimensional latent representation of the data for generating new samples. This group encompasses various models with varying objectives, including Variational Autoencoders (VAEs) [117], GANs, and, under specific assumptions, DMs.

- **Energy-Based Models:** Energy-based models define an energy function to measure the compatibility between the data and the model. These models aim to learn the

data distribution by minimizing the energy of real data points and maximizing the energy of generated samples.

In the following, we will primarily focus on ARMs, DMs, and GANs. GANs are the central subject of this thesis, while ARMs and DMs have been chosen for comparison as they currently demonstrate the best scalability among all deep generative model classes. For a more extensive treatment of other model classes, we refer the reader to [228].

2.1.1 Autoregressive Models

Autoregressive Models (ARMs) decompose the data distribution $p_{\text{data}}(\mathbf{x})$ into several less complex conditional distributions. ARMs treat $p_{\text{data}}(\mathbf{x})$ as a joint distribution that can be factorized via the product rule as follows

$$p_{\text{data}}(\mathbf{x}) = p(\mathbf{x}_0) \prod_{i=1}^{D} p(\mathbf{x}_i \mid \mathbf{x}_{<i})$$

where $\mathbf{x}_{<i}$ represents all $\mathbf{x}$'s up to i-th entry. In NLP, this decomposition is quite natural as it corresponds to token-by-token prediction [1]. For image synthesis, a model can predict pixel-by-pixel [37] or token-by-token in a pretrained latent space [67].

Modeling all conditional distributions $p(\mathbf{x}_i \mid \mathbf{x}_{<i})$ becomes intractable for long sequences. To address this, ARMs make a strong conditional independence assumption, where each variable's prediction depends only on a fixed number of previous variables rather than the entire history of the sequence. ARMs can be represented by deep neural networks, which use causal temporal convolutions to represent the variable's history more effectively. These models can also incorporate residual connections and dilated convolutions, which improve training dynamics and reduce the computational complexity to improve their performance [167]. An important model class is transformers [236], which uses self-attention layers instead of convolutions.

Advantages. ARMs provide multiple benefits, especially in NLP, where they play a crucial role. These models are optimized via a maximum-likelihood objective, which is stable and scalable. Also, their training is straightforward since they do not require backpropagation through time (BPTT) needed for training recurrent neural networks (RNNs) [94]. Furthermore, they enable flexible generation of diverse outputs at any stage during the sampling process. For instance, the model can formulate multiple plausible continuations for a given sequence. Similarly, for image generation, the model can first produce the upper half of the image (or is given a partial view of a real image) and then generate various distinct possibilities for the bottom half [35].

Disadvantages. The main drawback of autoregressive models is their inherent sequential nature. They produce data points in sequence, leading to slower synthesis and inference, particularly in high-dimensional spaces such as high-resolution images. This limitation can impact computational efficiency and hinder real-time applications.

[1] In NLP, a token refers to a meaningful unit of text such as a word, punctuation mark, or symbol.

Applications. Autoregressive models are widely employed in text [52, 27, 260], image [235, 37, 253], and audio synthesis [167, 238]. Currently, Parti [253] is the leading model for large-scale image synthesis. Parti generates high-fidelity, photorealistic images by approaching text-to-image generation as a sequence-to-sequence modeling problem. Utilizing a Transformer-based image tokenizer [252] and scaling the encoder-decoder transformer model to 20 billion parameters, Parti achieves state-of-the-art results in image synthesis tasks.

2.1.2 Diffusion Models

Diffusion-based generative models [219] (DMs) are a class of latent variable models, similar to Variational Autoencoders (VAEs) [117]. However, the concept that sets them apart is their unique approach to generating data - by reversing a diffusion process.

Following [112], we denote the data distribution by $p_{\text{data}}(\mathbf{x})$ with standard deviation σ_{data} and generate a family of mollified distributions $p(\mathbf{x};\sigma)$ by adding independent, identically distributed Gaussian noise of standard deviation σ to the data. When σ_{max} is significantly greater than σ_{data}, $p(\mathbf{x};\sigma_{max})$ is essentially Gaussian noise. For generating samples, we first randomly sample a noise image $\mathbf{x}_0$ from a normal distribution and sequentially denoise it into images $\mathbf{x}_i$ with decreasing noise levels from $\sigma_0 = \sigma_{max}$ to $\sigma_N = 0$. Each image $\mathbf{x}_i$ is distributed according to $p(\mathbf{x}_i;\sigma_i)$, and the final image $\mathbf{x}_N$ aligns with the data distribution.

The sequential nature of DMs opens up several interesting design spaces, including sampling decisions such as the selection of the solver and the determination of steps, training considerations like loss weighting and noise distribution, and aspects of network architecture like input and output scaling and conditioning. For a detailed treatment, we refer to [112].

Advantages. DMs are notably stable, robust, and relatively easy to train, making them suitable for fast experimentation and training at scale. Furthermore, they are recognized for their sample diversity and the ability to avoid mode collapse, a common problem with other generative models like GANs. While sampling from DMs can be time-consuming, the iterative process allows for continuous refinement of the results, leading to high-quality outputs. A method called *classifier-free guidance* [93], which trades off diversity for sample quality, plays a crucial role in achieving high-quality outputs with DMs and generally outperforms comparable methods like classifier guidance for DMs [54] or truncation for GANs [148].

Disadvantages. DMs utilize a sequential sampling process that significantly impacts their speed. While there has been notable progress in speeding up DMs [199, 112, 146], they still lag behind GANs regarding inference speed. Another disadvantage is that DMs maintain the dimensionality of the input throughout the entire model, lacking a bottleneck mechanism, similar flow-based models [119]. This characteristic prevents DMs from generating a compressed, low-dimensional representation of the data, which could be beneficial for downstream tasks and provide insights into the underlying structure of the data.

Applications. DMs have become increasingly prevalent in image synthesis. They are employed in both unconditional [91, 165, 221] and conditional [54, 164, 191, 194, 183] con-

texts. Several foundational models such as Dalle-2 [183], Imagen [194], and Stable Diffusion [191] demonstrate the capabilities of DMs when applied to image synthesis at scale. Building upon these foundational models, a typical application has been creating personalized DMs [193, 73, 156]. These models synthesize novel images of the same subject in various contexts and styles. Furthermore, DMs can be fine-tuned on smaller, task-specific datasets to produce high-quality samples. While a foundational model can generate almost anything imaginable, it might struggle with more specific subjects like human faces or hands. Fine-tuning on high-quality data allows for improved performance on these particular tasks but at the expense of the model's generality. In essence, quality is traded for generality. This fine-tuning process can be executed with just a fraction of the computational resources required for training the initial foundational model.

2.1.3 Generative Adversarial Networks

Generative Adversarial Networks (GANs) [79] employ a generator (G) and a discriminator (D) which compete with each other in a zero-sum game. This adversarial setup results in a model where the generator creates synthetic data resembling the real data, and the discriminator aims to distinguish between the real and generated data. GANs are often referred to as *implicit* models because they do not explicitly define the probability of the data distribution but learn to generate new samples through a learned transformation of a random noise vector. GAN training can be described as a two-player minimax game with a value function $V(G,D)$:

$$V(D,G) = \left(\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[\log(1-D(G(\mathbf{z})))]\right) \tag{2.1}$$

where $\mathbf{x}$ represents real data, $\mathbf{z}$ is a latent vector, $D(\mathbf{x})$ is the discriminator's estimate of the probability that $\mathbf{x}$ is real, and $G(\mathbf{z})$ is the generator's output given $\mathbf{z}$. A common approach to improve the fidelity of generated samples in GANs is the truncation trick [148, 25, 115]. This method limits the range of the latent space, moving sampled latents $\mathbf{z}$ towards the mean of the original distribution, effectively sacrificing diversity to enhance sample quality.

Advantages. The generator network produces outputs in a single forward pass rather than sequentially, making the generation process highly efficient and fast. The direct generation of samples also allows for the incorporation of additional losses to guide training. Examples include clip guidance for enhanced image-text alignment [205] and gaze loss to better match the gaze between a driving frame and neural head avatar [62]. GANs also establish a 'meaningful' latent space, where minor, linear alterations in the latent vector – referred to as a *latent walk* – yield plausible and semantically coherent changes in the output. This enables smooth sample transitions and is widely utilized in image editing tasks, where a latent walk is performed in a direction correlating to semantic modifications or alterations in viewpoint or lighting [1, 46, 173, 5, 233].

Disadvantages. GANs are notoriously hard to train, as they optimize a saddle-point problem, which is generally more challenging than local optimization problems [152]. The dual

optimization of generator and discriminator often introduces unstable dynamics, which leads to sub-optimal solutions. Commonly, regularization can stabilize the training [151, 153], which, however, can impair overall performance when scaling to larger datasets [24]. A common problem in GAN training is mode collapse, where the generator starts producing a limited variety of samples, thus collapsing multiple modes of the true data distribution into a single mode. Mode collapse limits the diversity of generated samples, and the model fails to represent the broader data distribution. Lastly, tracking the training progress is challenging as the losses are not informative of the current sample quality or diversity. The GAN loss function primarily represents the competition between the generator and discriminator rather than the quality of the generated data [22]. Consequently, a proxy metric, such as the Fréchet Inception Distance (FID) [90], is often used, which can exhibit blind spots regarding the perceptual quality of certain categories like human faces [128].

Applications. GANs have established themselves as the standard model for image synthesis on small to medium-sized unimodal datasets such as FFHQ [116] or LSUN [251]. They have also found widespread applications in image-to-image translation tasks [103, 170] and superresolution tasks [241, 240]. Besides image synthesis, GANs are commonly used for audio synthesis [122] and anomaly detection [55]. Real-time applications are another promising area for GANs, with the emergence of technologies like neural head avatars [62] and photorealism enhancement [188]. On large-scale image synthesis, models like BigGAN [24] have showcased the potential of GANs, although its impressive results have since been surpassed by DMs [54]. Interestingly, after the initial success of BigGAN, no bigger models have been trained, indicating the challenges associated with scaling up GANs.

2.2 The Basics of GAN Training

GANs are commonly trained in alternating fashion. We discuss the standard training algorithm and use a simple one-dimensional example to demonstrate how GANs function intuitively. Subsequently, we offer a brief discussion on the topic of regularization. For an in-depth review concerning different training approaches, adversarial losses, and regularization terms, we refer to [244].

Training Algorithm. The initial step in training GANs involves updating the discriminator D. First, an equal number B of real data points, $\mathbf{x}_b$, and latent vector samples, $\mathbf{z}_b$, are drawn. The latter is utilized to generate samples, $\hat{\mathbf{x}}_b = G(\mathbf{z}_b)$. Initially, Goodfellow et al. [78] propose to perform the discriminator update k times, yet, they set k to 1. Setting k to 1 is also common in practice, and we, therefore, assume a single discriminator step per generator step, simplifying the overall algorithm. After every iteration, the discriminator's weights $\mathbf{w}_D$ are updated through stochastic gradient ascent, according to

$$\nabla_{\mathbf{w}_D} \frac{1}{B} \sum_{b=1}^{B} \log D(\mathbf{x}_b) + \log(1 - D(G(\mathbf{z}_b))) \tag{2.2}$$

Following this, the generator receives an update. Fresh samples of the latent vector, $\mathbf{z}$,

are drawn, and the generator's weights $\mathbf{w}_G$ are updated via

$$\nabla_{\mathbf{w}_G} \frac{1}{B} \sum_{b=1}^{B} \log(1 - D(G(\mathbf{z}_b))) \tag{2.3}$$

The algorithm is terminated when the training converges; convergence is usually measured by independent proxy metrics such as FID, as mentioned in Section 2.1.3. The training algorithm is summarized in Algorithm 1.

Algorithm 1 Training GANs

while not converged **do**
  Draw B training samples $\{\mathbf{x}_1, \ldots, \mathbf{x}_B\}$ from $p_{data}(\mathbf{x})$
  Draw B latent samples $\{\mathbf{z}_1, \ldots, \mathbf{z}_B\}$ from $p(\mathbf{z})$
  Update the **discriminator** D by **ascending** its stochastic gradient:
    $\nabla_{\mathbf{w}_D} \frac{1}{B} \sum_{b=1}^{B} \log D(\mathbf{x}_b) + \log(1 - D(G(\mathbf{z}_b)))$

  Draw B latent samples $\{\mathbf{z}_1, \ldots, \mathbf{z}_B\}$ from $p(\mathbf{z})$
  Update the **generator** G by **descending** its stochastic gradient:
    $\nabla_{\mathbf{w}_G} \frac{1}{B} \sum_{b=1}^{B} \log(1 - D(G(\mathbf{z}_b)))$
end while

1D example. We can understand the training process of a GAN better by considering a simple example with a linear generator

$$\begin{aligned} \mathbf{x} &\sim \mathcal{N}(\mu, \sigma) \\ \mathbf{z} &\sim \mathcal{N}(0, 1) \\ G(z) &= w_0^G + w_1^G z \\ D(x) &= \sigma(w_0^D + w_1^D x + w_2^D x^2) \end{aligned}$$

The given data distribution and the prior are two 1D Gaussians and we initialize $G(z) = z$ and $D(x) = \sigma(x)$, i.e., $p_{model}(x) = \mathcal{N}(x|0,1)$. Our goal is to train $\mathbf{w}_G$ and $\mathbf{w}_D$ such that $p_{model}(x) = p_{data}(x) = \mathcal{N}(x|\mu, \sigma)$. We visualize the training process in this setup in Fig. 2.1.

Regularizing Training Dynamics. Arjovsky et al. [13] highlight that Lipschitz continuity of the discriminator with respect to its input space is essential for achieving stable training. Intuitively, Lipschitz continuity places a bound on how much the output of a function (in this case, the discriminator) can change in response to changes in the input. Formally, a discriminator D is said to be Lipschitz continuous if the following inequality holds

$$|D(\mathbf{x}_1) - D(\mathbf{x}_2)| \leq K|\mathbf{x}_1 - \mathbf{x}_2|, \tag{2.4}$$

where $K \geq 0$ is a Lipschitz constant and x_i are vectors in the input space. Enforcing Lip-

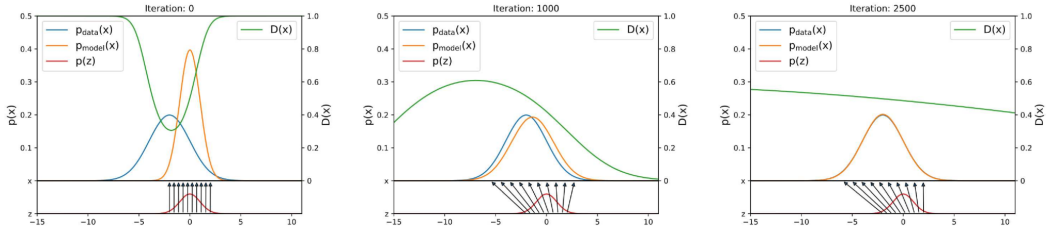

Figure 2.1: 1D example. *GANs are trained by simultaneously updating the discriminative distribution, $D(\mathbf{x})$, to discriminate between samples from the data generating distribution, p_{data}, and samples from the generative distribution, p_{model}. The lower horizontal line indicates the domain from which $\mathbf{z}$ is sampled, in this case, from a standard normal distribution. The horizontal line above represents part of the domain of x. The upward arrows illustrate how the mapping $\mathbf{x} = G(\mathbf{z})$ transforms the initial z distribution.* ***(a)*** *At the start of training, p_{data} and p_{model} have large non-overlapping regions.* ***(b)*** *After an update to G, the gradient of D has guided G(z) to flow to regions more likely to be classified as data.* ***(c)*** *After several steps of training, G and D reach a point where both cannot improve further as p_{data} equals p_{model}. At this stage, the discriminator cannot differentiate between the two distributions, resulting in $D(x) \approx 0.5$, indicating that D is equally likely to classify a sample as coming from either the data distribution or the model distribution. The example and figures are from [74].*

schitz continuity often involves gradient penalties. These penalties bound the magnitude of the gradients to ensure that they stay within a specified range. A common and effective regularization method for this purpose is the R1 Regularization [151], defined as:

$$\mathcal{L}_{R1} = \frac{\lambda}{2}\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\|\nabla D(\mathbf{x})\|^2] \tag{2.5}$$

Here, $\nabla D(\mathbf{x})$ is the gradient of the discriminator output with respect to its input. The term λ is a hyperparameter that regulates regularization strength. R1 regularization augments the GAN value function $V(G,D)$ as follows

$$V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\log D(\mathbf{x}) - \frac{\lambda}{2}\|\nabla D(\mathbf{x})\|^2\right] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[\log(1 - D(G(\mathbf{z})))] \tag{2.6}$$

R1 Regularization aims to prevent the discriminator from deviating from the Nash Equilibrium, where neither the generator nor the discriminator can improve their performance by changing their strategies by penalizing the gradient on real data alone. When the generator distribution aligns with the true data distribution and the discriminator evaluates to zero on the data manifold, the gradient penalty ensures that the discriminator cannot create a non-zero gradient orthogonal to the data manifold without incurring a loss.

Another way of ensuring Lipschitzness in the discriminator is normalizing its weights. Spectral Normalization [154] is the most widely used method for GAN training. This

method ensures Lipschitz continuity by constraining the spectral norm, the largest singular value of the weight matrix, of each layer in the discriminator network. This constraint effectively controls the Lipschitz constant.

2.3 StyleGAN: A Style-Based Generator Architecture

StyleGAN [115] emerged as a breakthrough in generative modeling, being the first to achieve impressive photorealism at high-resolution image synthesis. The distinctiveness of StyleGAN lies not only in its remarkable results but also in the unique aspects that define it: its peculiar architecture, as depicted in Fig. 2.2, and its well-engineered training strategy. Furthermore, StyleGAN exhibits characteristics beyond merely reproducing samples from the target distribution. Karras et al. [115] observed that StyleGAN establishes a remarkably structured latent space. This structure is achieved without any explicit supervisory signal. Such a structured, disentangled latent space can be leveraged for downstream tasks like image editing. While an expansive body of literature exists on leveraging StyleGAN for diverse editing tasks, our primary emphasis will be on its general architecture and training strategy. For a comprehensive review of downstream applications of StyleGAN, we direct readers to the survey by Bermano et al. [17].

2.3.1 Network Architecture

StyleGAN, adhering to the GAN setup, comprises a generator and a discriminator. Notably, it shifted the research focus from discriminator modifications [63, 158, 239] to generator design.

Generator. A StyleGAN generator comprises a mapping network $\mathbf{G}_m$ and a synthesis network $\mathbf{G}_s$. First, $\mathbf{G}_m$ maps a normally distributed latent code $\mathbf{z}$ to a style code $\mathbf{w}$. This conversion process is executed to align the probability of sampling a distinct combination of image attributes in the latent space with their respective frequencies in the dataset. This style code $\mathbf{w}$ is then used for modulating the convolution kernels of $\mathbf{G}_s$ to control the synthesis process. The synthesis network, $\mathbf{G}_s$, proceeds from a constant spatial input. The input undergoes a sequence of operations, including convolutions, non-linearities, and upsampling over N layers to generate an image. Fig. 2.2 provides an overview of the StyleGAN generator architectures in comparison to a standard GAN.

Discriminator. StyleGAN builds upon the architecture of its predecessor, ProgGAN [111], utilizing the same discriminator – a standard convolutional neural network (CNN). This discriminator features a unique minibatch standard deviation layer. This layer calculates batch statistics and feeds this information into the discriminator's penultimate layer, which can improve the diversity of the generated images. Karras et al. [116] thoroughly investigate different discriminator architectures and settle on a residual architecture. Despite numerous attempts to update the discriminator, including different architectural designs like a U-Net discriminator [207] and vision transformers [132], the standard StyleGAN2 discriminator remains the most popular choice due to its proven effectiveness and reliability across diverse applications.

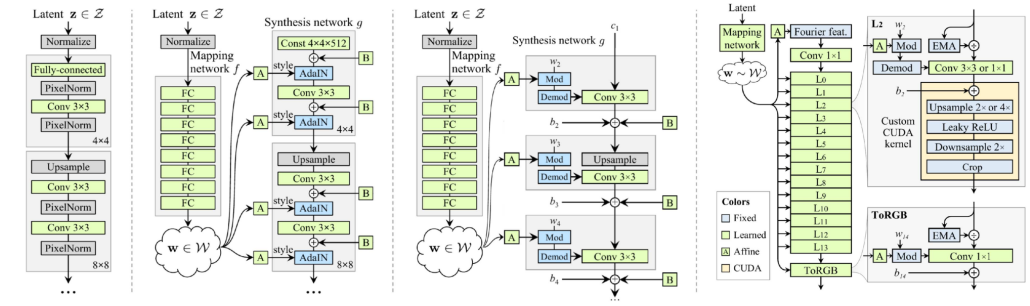

Figure 2.2: Comparative overview of the generator architecture evolution in the StyleGAN series. *(a) A standard GAN uses a single, direct mapping from the latent code* $\mathbf{z}$ *to the image. (b) StyleGAN introduces a two-step process with a mapping network* $\mathbf{G}_m$ *and a synthesis network* $\mathbf{G}_s$*. It first maps the input to an intermediate latent space W, controlling the generator through Adaptive Instance Normalization (AdaIN) at each convolution layer. Noise B is added after each convolution, before the nonlinearity. The mapping network consists of 8 layers and the synthesis network has 18 layers — two for each resolution from* 4×4 *to* 1024×1024*. (c) StyleGAN2 reduces artifacts by shifting normalization from individual feature maps to convolutional kernels. It introduces changes to the original architecture, including removing redundant operations at the beginning, moving the addition of bias* b_i *and noise B to be outside the active area of a style, and adjusting only the standard deviation per feature map. This revised architecture allows for the replacement of AdaIN with a demodulation operation applied to the weights of each convolution layer. (d) StyleGAN3 addresses the texture sticking problem using alias-free operations. The main path of this generator consists of Fourier features and normalization, modulated convolutions, and filtered nonlinearities. The figure is compiled of illustrations from [115, 116, 114].*

2.3.2 The Evolution of StyleGAN

Further advancements in StyleGAN's architecture, marked by iterations such as StyleGAN2 [116] and StyleGAN3 [114], continually address and reduce artifacts specific to the model, improving the overall quality of generated images.

In its initial architecture, StyleGAN adopts adaptive instance normalization (AdaIN) [97], which normalizes each channel of the feature maps to zero mean and unit variance, followed by a re-scaling process using new means and variances derived from the given latent code. One challenge encountered with AdaIN is the loss of information between different channels, leading to water droplet-like artifacts as shown in Fig. 2.3 (Left). Normalization can be shifted from individual feature maps to a modulation of the convolutional kernels to resolve this. The resulting normalization is based on expected feature statistics rather than exact signal strength, eradicating the need to obscure signal strength information and thus eliminating these artifacts. The redesigned generator architecture is shown in Fig. 2.2 (c).

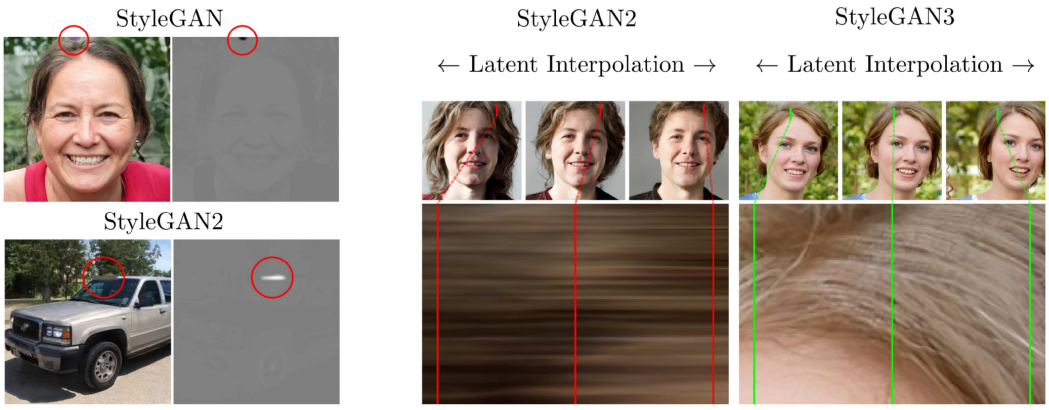

Figure 2.3: Artifacts in StyleGAN and StyleGAN2. *Left: Adaptive Instance Normalization (AdaIN) in StyleGAN leads to systemic water droplet-like artifacts in all generated images, evident in all feature maps from the 64x64 resolution onward. Right: An example of texture sticking. StyleGAN2 retains hair details at the same coordinates during a latent space interpolation, resulting in unwanted horizontal streaks. Ideally, the hair should animate smoothly, creating a time-varying field as demonstrated by the alias-free StyleGAN3. The visualizations are from [115, 116].*

StyleGAN3 [114] directly addresses the issue of *texture sticking,* a phenomenon where finer details, like hair or beard, appear rigidly tied to pixel coordinates, see Fig. 2.3 (Right). To mitigate this issue, StyleGAN3 introduces an architecture that promotes a more natural transformation hierarchy, such that the precise sub-pixel position of each feature is exclusively derived from the underlying coarse features. Towards this goal, StyleGAN3 employs alias-free primitive operations which ensure no preferred positions for the generated features, achieving translation equivariance. An upsampling and downsampling operation wraps each non-linearity to prevent aliasing. The low-pass filters used in these operations are designed to balance image quality, antialiasing, and training speed. While StyleGAN and StyleGAN2 utilize a learned constant, StyleGAN3 opts for constant Fourier features to enable direct control over a sample's rotation and translation. The StyleGAN3 generator is depicted in Fig. 2.2 (d).

2.3.3 Advanced Training Methods

In the training process of GANs, achieving stability presents a significant challenge. Equally important is the quality of the learned representation, which impacts the effectiveness of downstream tasks. For instance, a smoother latent space enables more efficient GAN inversion [116]. In the following, we highlight advanced techniques to train StyleGAN effectively. These methods complement the basic techniques, e.g., gradient penalties, introduced earlier in Section 2.2. The techniques below were introduced for StyleGAN training but broadly apply to GAN training in general. An exception is latent space regularization, which is specific to StyleGAN.

Progressive Growing. As proposed by Karras et al. [111], progressive growing is a GAN training method that incrementally increases image resolution by progressively adding layers to the generator and discriminator. This process allows the model to grasp the large-scale image structure initially before incrementally adding finer details. The generator and discriminator networks grow synchronously, with new layers smoothly blended in. This strategy enhances stability, speeds up convergence due to more efficient training of smaller networks at lower resolutions, and improves the overall sample quality.

Differentiable Augmentation. Differentiable Augmentation for GANs [113, 264, 234, 266] addresses the challenge of training GANs on small datasets. In the setting of small datasets, the discriminator easily overfits, which results in meaningless feedback to the generator [114]. These methods utilize a range of augmentations to prevent overfitting while ensuring that the augmentations do not leak into the generated images. Augmentation leakage is a common issue that deteriorates sample quality. A critical insight of these works is that the best results are achieved by applying augmentations to both real and generated images.

Regularizing the Latent Space. Style mixing and path length regularization are methods for regularizing style-based generators. In style mixing, an image is generated by independently feeding sampled style codes $\mathbf{w}$ into different layers of $\mathbf{G}_s$. Path length regularization encourages that a step of fixed size in latent space results in a corresponding fixed change in pixel intensity of the generated image [116]. This inductive bias leads to a smoother generator mapping and has numerous advantages, including fewer artifacts, more predictable training behavior, and better inversion.

2.4 Scaling Neural Networks

In this thesis, we initially investigate GANs in a small-scale context, employing single-GPU configurations and smaller datasets. We then move toward more complex multi-node, multi-GPU setups and handling datasets comprising hundreds of millions of data points. The following section provides a focused overview of the challenges inherent in scaling neural networks along three key dimensions: model size, computational capacity, and data volume.

2.4.1 Scaling Model Size

AlexNet [126] was the first to demonstrate the importance of larger datasets and models for the performance of neural networks. This demonstration led to a trend towards larger scales in machine learning, contributing to the development of various architectures. Among these, the Transformer [236] stands out, especially in the field of NLP, due to its flexibility stemming from its lack of inductive bias. GPT-3 [27] exemplifies how the model size and data scaling can enhance model generalization. Likewise, Vision Transformers (ViT) [57] have been developed for computer vision tasks. ConvNexts [145] have also affirmed the effectiveness of CNNs in this domain, an architecture that successfully competes with ViTs.

The development of these novel architectures has enabled scaling to more complex tasks than just classification, particularly when moving to text labels. A prime example of this shift is Contrastive Language-Image Pre-Training (CLIP) [178]. CLIP demonstrates the capabilities of large-scale models in dealing with tasks that require both vision and language processing.

Normalization techniques are a critical building block of modern neural network architecture and are similarly essential for successfully training GANs. While efficient, BatchNorm [102] presents challenges in large-scale neural network training due to the computational demands of synchronizing batch statistics during distributed training. Therefore, LayerNorm [15] and GroupNorm [246] are commonly preferred for their superior scalability. However, within GANs' discriminator architectures, BatchNorm is often employed to enable the discriminator to process batch information effectively. The Minibatch Standard Deviation Layer [111] can constitute a viable alternative.

2.4.2 Scaling Compute

Distributed training is essential for handling large datasets, speeding up training time, and accommodating models too large to fit into the memory of a single machine. Furthermore, it offers scalability, permitting the utilization of additional resources as datasets expand and models grow. Two primary methods for distributing neural network training across multiple accelerators, such as GPUs or TPUs [108], are model parallelism and data parallelism [255].

- **Data Parallelism.** In data parallelism, a copy of the model is placed on each device, and each device computes the output and the model's gradient for a unique subset of the data. Before updating the model, the gradients are combined across all devices, typically through averaging. This method is commonly used when the model fits on a single device.

- **Model Parallelism.** In model parallelism, different parts of the model are placed on different devices, and each device computes the output and the gradient of its part of the model. This method is typically used when the model is too large to fit on a single device [253]. The communication between devices can become a bottleneck, as different parts of the model must exchange intermediate results.

Various strategies have been proposed to mitigate the communication bottleneck introduced by model parallelism. GPipe [99], for instance, pipelines sub-sequences of layers across different accelerators. This method reduces the communication overhead and allows computation and communication to overlap, using the available computational resources more effectively. Data Parallelism and model parallelism can be combined by splitting the model and the data across multiple devices, known as hybrid parallelism. In this work, we will leverage data parallelism since our biggest model is still small enough to fit on a single GPU; hence, there is no need to introduce additional communication overhead through model parallelism.

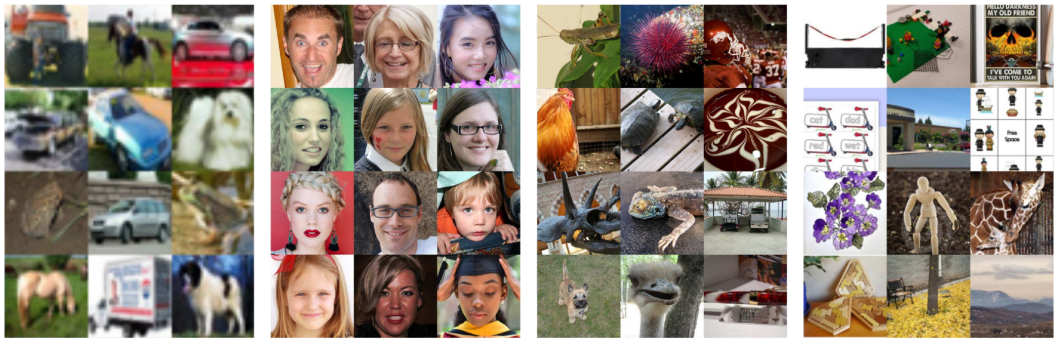

Figure 2.4: Evolving Complexity in Generative Modeling Datasets. *Shown are sample images from a selection of datasets used in generative modeling research. From left to right: CIFAR10 (2009) [125], FFHQ (2020) [116], ImageNet (2009) [48], and LAION-5B (2022) [208]. These samples underscore the increasing complexity trend in datasets over time. Despite FFHQ being introduced later than ImageNet, it's noteworthy that generative models producing sample quality on par with those trained on FFHQ only appeared recently [53, 92].*

2.4.3 Scaling Datasets

Machine learning datasets, particularly in computer vision, NLP, and generative modeling, exhibit an ongoing trend toward increased size and complexity. ImageNet [48] is a notable milestone in computer vision, instigating a series of subsequent advancements. Proprietary datasets such as Instagram-1B [249] and JFT300M [249] have been assembled for class-conditional pre-training at a larger scale.

Following the efforts to compile large image datasets with class labels, the focus has expanded toward creating image-text datasets. Pioneering works such as MS-COCO [140] and Visual Genome [124], which leverage human annotation, have produced high-quality labels, albeit limited to 330K and 5M examples, respectively. The web-harvested YFCC-100M dataset [227] broadens the scale to about 99 million images and one million videos. Further enhancements, such as the Conceptual Captions dataset (CC3M) [211], and its successor CC12M [34] utilize web-collected images and alt-text, incorporating additional data cleaning. An open-source effort, LAION-5B [208], amassed 5.85 billion CLIP-filtered image-text pairs, with 2.32B in English, while the remaining are multilingual. The latest open-source large-scale dataset, DataComp [71], provides 12.8 billion image-text features and a multi-scale design that facilitates studying scaling trends and provides accessibility to researchers with varying resources.

Generative modeling research progressively shifted towards more complex datasets, from CIFAR10 [125], a dataset of 60,000 samples at a 32x32 pixel resolution, to FFHQ [116], a high-resolution (1024x1024 pixels) face dataset with 70,000 samples. Recent advancements in diffusion models demonstrate impressive performance on large and diverse datasets like ImageNet and LAION-5B, effectively utilizing data at scale. Figure 2.4 displays a selection of dataset samples, underlining the increasing complexity of datasets over time.

3 Leveraging Pretrained Feature Networks for GAN Training

3.1 Introduction

In GAN training, the generator's task is to generate an RGB image; the discriminator aims to distinguish real from fake samples. On closer inspection, the discriminator's task is two-fold: First, it projects the real and fake samples into a meaningful space, i.e., it learns a representation of the input space. Second, it discriminates based on this representation. Unfortunately, training the discriminator jointly with the generator is a notoriously hard task. While discriminator regularization techniques help to balance the adversarial game [127], standard regularization methods like gradient penalties [150] are susceptible to hyperparameter choices [115] and can lead to a substantial decrease in performance [24].

In this chapter, we explore the utility of pretrained representations to improve and stabilize GAN training. Using pretrained representations has become ubiquitous in computer vision [184, 121, 123] and natural language processing [181, 175, 96]. While combining pretrained perceptual networks [217] with GANs for image-to-image translation has led to impressive results [223, 239, 188, 66], this idea has not yet materialized for unconditional noise-to-image synthesis. Indeed, we confirm that a naïve application of this idea does not lead to state-of-the-art results as strong pretrained features enable the discriminator to dominate the two-player game, resulting in vanishing gradients for the generator [11]. In the following, we demonstrate how these challenges can be overcome and identify two key components for exploiting the full potential of pretrained perceptual feature spaces for GAN training: **feature pyramids** to enable multi-scale feedback with multiple discriminators and **random projections** to better utilize deeper layers of the pretrained network.

We conduct extensive experiments on small and large datasets with a resolution of up to 1024^2 pixels. Across all datasets, we demonstrate state-of-the-art image synthesis results at significantly reduced training time. We also find that Projected GANs increase data efficiency and avoid the need for additional regularization, rendering expensive hyperparameter sweeps unnecessary.

We categorize related work into two main areas: utilizing pretrained representations for GAN training and discriminator design.

Pretrained Models for GAN Training. Work on leveraging pretrained representations for GANs can be divided into two categories: First, transferring parts of a GAN to a new dataset [155, 85, 242, 263] and, second, using pretrained models to control and improve GANs. The latter is advantageous as pretraining does not need to be adversarial. Our work

falls into this second category. Pretrained models can be used as a guiding mechanism to disentangle causal generative factors [204], for text-driven image manipulation [171], matching the generator activations to inverted classifiers [215, 98], or to generate images via gradient ascent in the latent space of a generator [163]. The non-adversarial approach of [200] learns generative models with moment matching in pretrained models; however, the results remain far from competitive to standard GANs. An established method is the combination of adversarial and perceptual losses [106]. Commonly, the losses are combined additively [59, 239, 197, 131, 66]. Additive combination, however, is only possible if a reconstruction target is available, e.g., in paired image-to-image translation settings [269]. Instead of providing the pretrained network with a reconstruction target, Sungatullina et al. [223] propose to optimize an adversarial loss on frozen VGG features [217]. They show that their approach improves CycleGAN [269] on image translation tasks. In a similar vein, [188] recently proposed a different perceptual discriminator. They utilize a pretrained VGG and connect its features with the prediction of a pretrained segmentation network. The combined features are fed into multiple discriminators at different scales. The two last approaches are specific to the image-to-image translation task. We demonstrate that these methods do not work well for the more challenging unconditional setting where the entire image content is synthesized from a random latent code.

Discriminator Design. Much work on GANs focuses on novel generator architectures [24, 115, 116, 256], while the discriminator often remains close to a vanilla convolutional neural network or mirrors the generator. Notable exceptions are [262, 207], which utilize an encoder-decoder discriminator architecture. However, in contrast to us, they neither use pretrained features nor random projections. A different line of work considers a setup with multiple discriminators applied to either the generated RGB image [64, 56] or low-dimensional projections thereof [162, 7]. The use of several discriminators promises improved sample diversity, training speed, and training stability. However, these approaches are not utilized in current state-of-the-art systems because of diminishing returns compared to the increased computational effort. Providing multi-scale feedback with one or multiple discriminators has been helpful for both image synthesis [111, 110] and image-to-image translation [239, 170]. While these works interpolate the RGB image at different resolutions, our findings indicate the importance of multi-scale *feature maps*, showing parallels to the success of pyramid networks for object detection [139]. Lastly, to prevent overfitting of the discriminator, differentiable augmentation methods have recently been proposed [113, 264, 234, 266]. We find that adopting these strategies helps exploit the full potential of pretrained representations for GAN training.

3.2 Projected GANs

3.2.1 Definition

As we described in Chapter 2, GANs aim to model the distribution of a given training dataset. A generator G maps latent vectors $\mathbf{z}$ sampled from a simple distribution $p_{\mathbf{z}}$ (typically a normal distribution) to corresponding generated samples $G(\mathbf{z})$. The discriminator D

then aims to distinguish real samples $\mathbf{x} \sim p_{\text{data}}$ from the generated samples $G(\mathbf{z}) \sim p_{G(\mathbf{z})}$. This basic idea results in the following minimax objective

$$\min_G \max_D \Big(\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}} \log(1 - D(G(\mathbf{z})))] \Big) \tag{3.1}$$

We introduce a set of feature projectors $\{P_l\}$, which map real and generated images to the discriminator's input space. Projected GAN training can thus be formulated as follows

$$\min_G \max_{\{D_l\}} \sum_{l\in\mathcal{L}} \Big(\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D_l(P_l(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[\log(1 - D_l(P_l(G(\mathbf{z}))))] \Big) \tag{3.2}$$

where $\{D_l\}$ is a set of independent discriminators operating on different feature projections. Note that we keep $\{P_l\}$ fixed in (3.2) and only optimize the parameters of G and $\{D_l\}$. The feature projectors $\{P_l\}$ should satisfy two necessary conditions: they should be differentiable and provide sufficient statistics of their inputs, i.e., they should preserve important information. Moreover, we aim to find feature projectors $\{P_l\}$ which turn the (difficult to optimize) objective in (3.1) into an objective more amenable to gradient-based optimization. We now show that a Projected GAN indeed matches the distribution in the projected feature space before specifying the details of our feature projectors.

3.2.2 Consistency

The Projected GAN objective in (3.2) no longer optimizes directly to match the true distribution p_T. To understand the training properties under ideal conditions, we consider a more generalized form of the consistency theorem of [162]:

Theorem 1. *Let p_T denote the density of the true data distribution and p_G the density of the distribution the Generator G produces. Let $P_l \circ T$ and $P_l \circ G$ be the functional composition of the differentiable and fixed function P_l and the true/generated data distribution, and* $\mathbf{y}$ *be the transformed input to the discriminator. For a fixed G, the optimal discriminators are given by*

$$D^*_{l,G}(\mathbf{y}) = \frac{p_{P_l\circ T}(\mathbf{y})}{p_{P_l\circ T}(\mathbf{y}) + p_{P_l\circ G}(\mathbf{y})}$$

for all $l \in \mathcal{L}$. In this case, the optimal G under (3.2) is achieved iff $p_{P_l\circ T} = p_{P_l\circ G}$ for all $l \in \mathcal{L}$.

In the following, we first prove Theorem 1 for a deterministic projection. The second proof demonstrates the theorem's validity when training with stochastic differentiable augmentations. From the theorem, we conclude that a feature projector P_l with its associated discriminator D_l encourages the generator to match the true distribution along the marginal through P_l. Therefore, at convergence, G matches the generated and true distributions in feature space.

Proof (deterministic). The following proof follows the consistency proofs in [162] and [79]. Let $\{P_l\}_{l\in\mathcal{L}}$ be a set of fixed feature projectors. Furthermore, let p_T be the density of

the true data distribution and p_G the density of the distribution the generator G produces. As in Theorem 1, $P_l \circ T$ and $P_l \circ G$ are functional compositions of P_l and the true/generated data distribution. The minimax objective in (3.2) is then defined via

$$\min_G \max_{\{D_l\}} \sum_{l \in \mathcal{L}} V_l(D_l, G) \tag{3.3}$$

where

$$\begin{aligned} V_l(D_l, G) &= \mathbb{E}_{\mathbf{x} \sim p_T}\left[\log D_l\left(P_l(\mathbf{x})\right)\right] + \mathbb{E}_{\mathbf{x} \sim p_G}\left[\log\left(1 - D_l\left(P_l(\mathbf{x})\right)\right)\right] \\ &= \mathbb{E}_{\mathbf{y} \sim p_{P_l \circ T}}\left[\log D_l(\mathbf{y})\right] + \mathbb{E}_{\mathbf{y} \sim p_{P_l \circ G}}\left[\log\left(1 - D_l(\mathbf{y})\right)\right] \\ &= \int_{\mathbf{y}} p_{P_l \circ T}(\mathbf{y}) \log(D_l(\mathbf{y})) + p_{P_l \circ G}(\mathbf{y}) \log(1 - D_l(\mathbf{y})) d\mathbf{y} \end{aligned} \tag{3.4}$$

In the following, we derive the optimal discriminator for a fixed G. For any $(a,b) \in \mathbb{R}^2 \setminus \{(0,0)\}$, the function $t \to a\log(t) + b\log(1-t)$ obtains its maximum in $[0,1]$ at $\frac{a}{a+b}$. Since the discriminators do not need to be defined outside $\operatorname{supp}(p_{P_l \circ T}) \cup \operatorname{supp}(p_{P_l \circ G})$, the maximum $\max_{\{D_l\}} V_l(D_l, G)$ is achieved for

$$D^*_{l,G}(y) = \frac{p_{P_l \circ T}(\mathbf{y})}{p_{P_l \circ T}(\mathbf{y}) + p_{P_l \circ G}(\mathbf{y})} \tag{3.5}$$

where G is fixed. Assuming a perfect discriminator, the minimax objective can be reformulated as

$$C(G) = \max_{\{D_l\}} \sum_l V_l(G, D_l) = \sum_l V_l(G, D^*_{l,G}) \tag{3.6}$$

Following the arguments of [79] and in [162], we obtain

$$C(G) = -2|\mathcal{L}|\log(2) + 2\sum_l JSD(p_{P_l \circ T} || p_{P_l \circ G}) \tag{3.7}$$

where JSD is the Jensen-Shannon divergence. Since the Jensen-Shannon divergence is non-negative and zero only in case of equality, the minimum is achieved iff $p_{P_l \circ T} = p_{P_l \circ G}$ for all l. This shows, that we achieve $\min_G C(G)$ iff $p_{P_l \circ T} = p_{P_l \circ G}$ for all l.

Proof (stochastic). We now show that the result above still holds when applying stochastic differentiable augmentations before the feature projections.

Utilizing a stochastic augmentation $f_{\theta,l}$ before the projection through P_l, can be viewed as a functional composition, i.e., $P_{\theta,l} = P_l \circ f_{\theta,l}$. The parameter $\theta \sim p_\Theta$ encompasses both the probability of applying the augmentation and its parameters, e.g., translation direction and magnitude. As in the deterministic case, the minimax objective is defined as

$$\min_G \max_{\{D_l\}} \sum_{l \in \mathcal{L}} V_l(D_l, G) \tag{3.8}$$

where

$$\begin{aligned} V_l(D_l, G) &= \mathbb{E}_{\mathbf{x}\sim p_T}\left[\mathbb{E}_{\theta\sim p_\Theta}[\log D_l(P_{\theta,l}(\mathbf{x}))]\right] + \mathbb{E}_{\mathbf{x}\sim p_G}\left[\mathbb{E}_{\theta\sim p_\Theta}[\log(1 - D_l(P_{\theta,l}(\mathbf{x})))]\right] \\ &= \mathbb{E}_{\theta\sim p_\Theta}\left[\mathbb{E}_{\mathbf{x}\sim p_T}[\log D_l(P_{\theta,l}(\mathbf{x}))] + \mathbb{E}_{\mathbf{x}\sim p_G}[\log(1 - D_l(P_{\theta,l}(\mathbf{x})))]\right] \\ &= \mathbb{E}_{\theta\sim p_\Theta}\left[\mathbb{E}_{\mathbf{y}\sim p_{P_{\theta,l}\circ T}}[\log D_l(\mathbf{y})] + \mathbb{E}_{\mathbf{y}\sim p_{P_{\theta,l}\circ G}}[\log(1 - D_l(\mathbf{y}))]\right] \\ &= \mathbb{E}_{\theta\sim p_\Theta}\left[\int_{\mathbf{y}} p_{P_{\theta,l}\circ T}(\mathbf{y})\log(D_l(\mathbf{y})) + p_{P_{\theta,l}\circ G}(\mathbf{y})\log(1 - D_l(\mathbf{y}))d\mathbf{y}\right] \\ &= \int_{\mathbf{y}} \mathbb{E}_{\theta\sim p_\Theta}[p_{P_{\theta,l}\circ T}(\mathbf{y})]\log(D_l(\mathbf{y})) + \mathbb{E}_{\theta\sim p_\Theta}[p_{P_{\theta,l}\circ G}(\mathbf{y})]\log(1 - D_l(\mathbf{y}))d\mathbf{y} \end{aligned} \tag{3.9}$$

Using the same arguments as above, we obtain that the maximum $\max_{\{D_l\}} V_l(D_l, G)$ is achieved for

$$D^*_{l,G}(y) = \frac{\mathbb{E}_{\theta\sim p_\Theta}[p_{P_{\theta,l}\circ T}(\mathbf{y})]}{\mathbb{E}_{\theta\sim p_\Theta}[p_{P_{\theta,l}\circ T}(\mathbf{y})] + \mathbb{E}_{\theta\sim p_\Theta}[p_{P_{\theta,l}\circ G}(\mathbf{y})]} \tag{3.10}$$

where G is fixed. Note that $\mathbb{E}_{\theta\sim p_\Theta}[p_{P_{\theta,l}\circ T}]$ and $\mathbb{E}_{\theta\sim p_\Theta}[p_{P_{\theta,l}\circ G}]$ are densities. Similar to above, we obtain $\min_G C(G)$ iff $\mathbb{E}_{\theta\sim p_{\Theta,l}}[p_{P_l\circ T}] = \mathbb{E}_{\theta\sim p_\Theta}[p_{P_{\theta,l}\circ G}]$ for all l.

3.2.3 Multi-Scale Discriminators

We obtain features from four layers L_l of a pretrained feature network F at resolutions $(L_1 = 64^2, L_2 = 32^2, L_3 = 16^2, L_4 = 8^2)$. We associate a separate discriminator D_l with the features at layer L_l, respectively. Each discriminator D_l uses a simple convolutional architecture with spectral normalization [153] at each convolutional layer. We observe better performance if all discriminators output logits at the same resolution (4^2). Accordingly, we use fewer down-sampling blocks for lower-resolution inputs. Following common practice, we sum all logits to compute the overall loss. For the generator pass, we sum the losses of all discriminators. More complex strategies [64, 7] did not improve performance in our experiments.

3.2.4 Random Projections

We observe that features at deeper layers are significantly harder to cover, as evidenced by our experiments in Section 3.3.1. We hypothesize that a discriminator can focus on a subset of the feature space while wholly disregarding other parts. This problem might be especially prominent in the deeper, more semantic layers. Therefore, we propose two different strategies to dilute prominent features, encouraging the discriminator to utilize all available information equally. Common to both strategies is that they mix features using differentiable random projections which are fixed, i.e., after random initialization, the parameters of these layers are not trained.

Cross-Channel Mixing (CCM). Empirically, we found two properties to be desirable: (i) the random projection should be information preserving to leverage the full representational power of F, and (ii) it should not be trivially invertible. The easiest way to mix across channels is a 1×1 convolution. A 1×1 convolution with an equal number of output

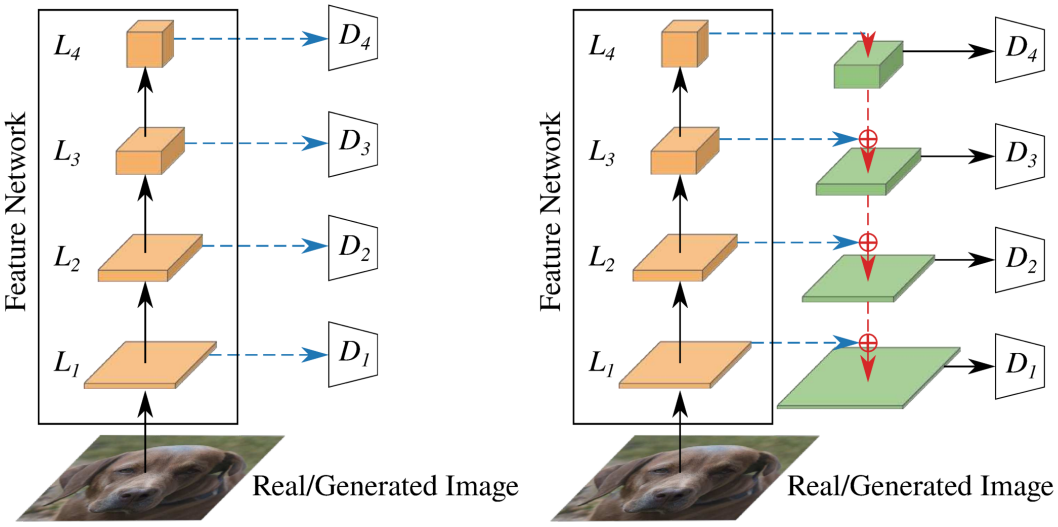

Figure 3.1: CCM and CSM. *Left: CCM (dashed blue arrows) employs 1×1 convolutions with random weights. Right: CSM (dashed red arrows) adds random 3×3 convolutions and bilinear upsampling, yielding a U-Network.*

and input channels is a generalization of a permutation [119] and consequently preserves information about its input. In practice, we find that more output channels lead to better performance as the mapping remains injective and, therefore, information preserving. Kingma et al. [119] initialize their convolutional layers as a random rotation matrix as a good starting point for optimization. We do not find this to improve GAN performance, arguably, since it violates (ii). We, therefore, randomly initialize the weights of the convolutional layer via Kaiming initialization [88]. Note that we do not add any activation functions. We apply this random projection at each of the four scales and feed the transformed feature to the discriminator as depicted in Fig. 3.1.

Cross-Scale Mixing (CSM). To encourage feature mixing *across* scales, CSM extends CCM with random 3×3 convolutions and bilinear upsampling, yielding a U-Net [192] architecture, see Fig. 3.1. However, our CSM block is simpler than a vanilla U-Net [192]: we only use a single convolutional layer at each scale. As for CCM, we utilize Kaiming initialization for all weights.

3.2.5 Pretrained Feature Networks

We ablate over varying feature networks. First, we investigate different versions of EfficientNets, which allow for direct control over model size versus performance. EfficientNets are image classification models trained on ImageNet [49] and designed to provide favorable accuracy-compute tradeoffs. Second, we use ResNets of varying sizes. To analyze the dependency on ImageNet features (Section 3.3.1), we also consider R50-CLIP [179], a ResNet optimized with a contrastive language-image objective on a dataset of 400 million (image, text) pairs. Lastly, we utilize a vision transformer architecture (ViT-Base)

[58] and its efficient follow-up (DeiT-small distilled) [229]. We do not choose an inception network [224] to avoid strong correlations with the evaluation metric FID [90]. We also evaluate several other neural and non-neural metrics to rule out correlations: Kernel Inception Distance (KID) [19], Precision and Recall [196], SwAV-FID [159], CLIP-FID [179] & Virtex-FID [50], and Sliced Wasserstein Distance (SWD) [111]. For a discussion of each metric, we refer to their respective papers.

3.3 Experiments

3.3.1 Ablation Study

To determine the best configuration of discriminators, mixing strategy, and pretrained feature network, we conduct experiments on LSUN-Church [251], which is medium-sized (126k images) and reasonably visually complex, using a resolution of 256^2 pixels. For the generator G we use the generator architecture of FastGAN [142], consisting of several upsampling blocks with additional skip-layer-excitation blocks. Using a hinge loss [137], we train with a batch size of 64 until 1 million real images have been shown to the discriminator, a sufficient amount for G to reach values close to convergence. If not specified otherwise, we use an EfficientNet-Lite1 [225] feature network in this section. We found that discriminator augmentation [113, 264, 234, 266] consistently improves the performance of all methods and is required to reach state-of-the-art performance. We leverage differentiable data augmentation [264], which we found to yield the best results in combination with FastGAN.

Which feature network layers are most informative? We first investigate the relevance of independent multi-scale discriminators. For this experiment, we do not use feature mixing. To measure how well G fits a particular feature space, we employ the Fréchet Distance (FD) [61] on the spatially pooled features denoted as FD_i for layer i. FDs across different feature spaces are not directly comparable. Therefore, we train a GAN baseline with a standard RGB discriminator, record FD_i^{RGB} at each layer and quantify the relative improvement via the fraction $\textit{rel-}FD_i = FD_i / FD_i^{RGB}$. We also investigate a perceptual discriminator [223], where feature maps are fed into different layers of the *same* discriminator to predict a single logit.

The results in Table 3.1 (No Projection) show that two discriminators are better than one and improve over the vanilla RGB baseline. Surprisingly, adding discriminators at deep layers hurts performance. We conclude that these more semantic features do not respond well to direct adversarial losses. We also experimented with discriminators at resized versions of the original image but could not find a setting of hyperparameters and architectures that improves over the single image baseline. Omitting the discriminators on the shallow features decreases performance, which is anticipated, as these layers contain most of the information about the original image. A similar effect has been observed for feature inversion [60] – the deeper the layer, the harder it is to reconstruct its input. Lastly, we observe that independent discriminators outperform the perceptual discriminator by a significant margin.

Discriminator(s)	$rel\text{-}FD_1 \downarrow$	$rel\text{-}FD_2 \downarrow$	$rel\text{-}FD_3 \downarrow$	$rel\text{-}FD_4 \downarrow$	$rel-FID \downarrow$
No Projection					
on L_1	0.56	0.32	0.31	0.55	0.66
on L_1, L_2	**0.35**	**0.21**	**0.23**	**0.47**	**0.53**
on L_1, L_2, L_3	0.42	0.26	0.28	0.64	0.90
on L_1, L_2, L_3, L_4	0.46	0.34	0.38	0.79	1.15
on L_2, L_3, L_4	0.95	0.67	0.71	1.19	1.99
on L_3, L_4	2.14	1.41	1.18	1.99	3.46
on L_4	10.92	5.74	2.56	2.79	5.08
Perceptual D	2.98	1.76	1.20	1.89	2.73
CCM					
on L_1	**0.27**	0.21	0.26	0.50	0.59
on L_1, L_2	**0.27**	**0.18**	**0.21**	**0.41**	**0.48**
on L_1, L_2, L_3	0.31	0.25	0.24	0.54	0.67
on L_1, L_2, L_3, L_4	0.53	0.34	0.34	0.59	0.77
Perceptual D	5.33	3.06	2.14	1.09	4.77
CCM + CSM					
on L_1	0.34	0.25	0.19	0.35	0.44
on L_1, L_2	**0.21**	0.18	0.16	0.27	0.31
on L_1, L_2, L_3	0.41	0.26	0.17	0.23	0.29
on L_1, L_2, L_3, L_4	0.26	**0.16**	**0.13**	**0.16**	**0.24**
Perceptual D	2.53	1.37	0.89	0.43	2.13

Table 3.1: Feature Space Fréchet Distances. *We aim to find the best combination of discriminators and random projections to fit the distributions in feature network F. We show the relative FD at different layers of F ($rel\text{-}FD_i$) between 50k generated and real images on LSUN-Church. $rel\text{-}FD_i$ is **normalized** using the baseline Fréchet Distances for a model with a standard single RGB image discriminator. Hence, values > 1 indicate worse performance than the RGB baseline. We report $rel\text{-}FD$ for four layers of an EfficientNet (L_1, L_2, L_3 and L_4 from shallow to deep), as well as relative Fréchet Inception Distance (FID) [90]. Note that $rel\text{-}FD_i$ should not be compared between different feature spaces, i.e., only within-column comparisons are meaningful. Blue boxes highlight the layers which we supervise via independent discriminators. The green box corresponds to a perceptual discriminator [223], which takes in all feature maps at once.*

	EfficientNet					ResNet			Transformer	
	lite0	lite1	lite2	lite3	lite4	R18	R50	R50-CLIP	DeiT	ViT
Params (M) ↓	2.96	3.72	4.36	6.42	11.15	11.18	23.51	23.53	92.36	317.52
IN top-1 ↑	75.48	76.64	77.47	79.82	81.54	69.75	79.04	N/A	85.42	85.16
FID ↓	2.53	1.65	1.69	1.79	2.35	4.16	4.40	3.80	2.46	12.38

***Table 3.2: Pretrained Feature Networks Study**. We train the projected GAN with different pretrained feature networks. We find that compact EfficientNets outperform both ResNets and Transformers.*

How can we best utilize the pretrained features? Given the insights from the previous section, we aim to improve the utilization of deep features. For this experiment, we only investigate configurations that include discriminators at high resolutions. Table 3.1 (CCM and CCM + CSM) presents the results for both mixing strategies. CCM moderately decreases the FDs across all settings, confirming our hypothesis that mixing channels results in better feedback for the generator. When adding CSM, we achieve another notable improvement across all configurations. Especially *rel-*FD_i at deeper layers are significantly decreased, demonstrating CSM's usefulness to leverage deep semantic features. Interestingly, we observe that the best performance is now obtained by combining all four discriminators. A perceptual discriminator is again inferior to multiple discriminators. We remark that integrating the original image via an independent discriminator, CCM, or CSM always resulted in worse performance. This failure suggests that naïvely combining non-projected with projected adversarial optimization impairs training dynamics.

Which feature network architecture is most effective? Using the best setting determined by the experiments above (CCM + CSM with four discriminators), we study the effectiveness of various perceptual feature network architectures for Projected GAN training. To ensure convergence, also for larger architectures, we train for 10 million images. Table 3.2 reports the FIDs achieved on LSUN-Church. Surprisingly, we find that there is no correlation with ImageNet accuracy. On the contrary, we observe lower FIDs for smaller models (e.g., EfficientNets-lite). This observation indicates that a more *compact* representation is beneficial while at the same time reducing computational overhead and, consequently, training time. R50-CLIP slightly outperforms its R50 counterpart, indicating that ImageNet features are not required to achieve low FID. For the sake of completeness, we also train with randomly initialized feature networks, which, however, converge to much higher FID values. In the following, we thus use EfficientNet-Lite1 as our feature network.

3.3.2 Quantitative Comparison to State-of-the-Art

This section conducts a comprehensive analysis demonstrating the advantages of Projected GANs with respect to state-of-the-art models. Our experiments are structured into three sections: evaluation of convergence speed and data efficiency (5.1) and comparisons on large (5.2) and small (5.3) benchmark datasets. We cover a wide variety of datasets in terms of size (hundreds to millions of samples), resolution (256^2 to 1024^2), and visual complexity (clip art, paintings, and photographs).

Evaluation Protocol. We measure image quality using the Fréchet Inception Distance (FID) [90]. Following [115, 116], we report the FID between 50k generated and all real images. We select the snapshot with the best FID for each method. In addition to image quality, we include a metric to evaluate convergence. As in [113], we measure training progress based on the number of real images shown to the discriminator (*Imgs*). We report the number of images required by the model for the FID to reach values within 5% of the best FID over training. Unless otherwise specified, we follow the evaluation protocol of [100] to facilitate fair comparisons. Specifically, we compare all approaches given the same fixed number of images (10 million). With this setting, each experiment takes roughly 100-200 GPU hours on an NVIDIA V100.

Baselines. We use StyleGAN2-ADA [113] and FastGAN [142] as baselines. StyleGAN2-ADA is the strongest model on most datasets in terms of sample quality, whereas FastGAN excels in training speed. We implement these baselines and our Projected GANs within the codebase provided by the authors of StyleGAN2-ADA [113]. For each model, we ran two kinds of data augmentation: differentiable data-augmentation [264] and adaptive discriminator augmentation [113]. We select the better-performing augmentation strategy per model. For all baselines and datasets, we perform data amplification through x-flips. Projected GANs use the same generator and discriminator architecture and training hyper-parameters (learning rate and batch size) for all experiments. For high-resolution image generation, additional upsampling blocks are included in the generator to match the desired output resolution. We carefully tune all hyper-parameters for both baselines for best results: we find that FastGAN is sensitive to the choice of batch size and StyleGAN2-ADA to the learning rate and R1 penalty.

Convergence Speed and Data Efficiency Following [100] and [254], we analyze the training properties of Projected GANs on LSUN-Church at an image resolution of 256^2 pixels and on the 70k CLEVR dataset [107]. In this section, we also train longer than 10 M images if necessary, as we are interested in convergence properties.

Convergence Speed. We apply Projected GAN training for both the style-based generator of StyleGAN2 and the standard generator with a single input noise vector of FastGAN. As shown in Fig. 3.2 (left), FastGAN converges quickly but saturates at a high FID. StyleGAN2 converges more slowly (88 M images) but reaches a lower FID. Projected GAN training improves both generators. Particularly for FastGAN, improvements in both convergence speed and final FID are significant, while improvements for StyleGAN2 are less pronounced. Remarkably, *Projected FastGAN* reaches the previously best FID of Style-

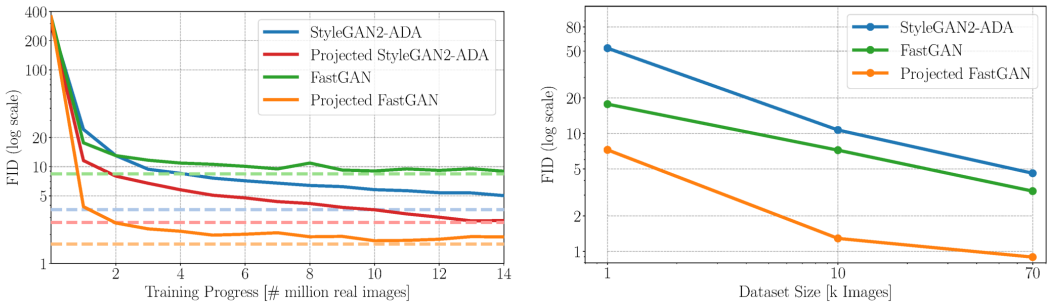

Figure 3.2: Training Properties. *Left: Projected FastGAN surpasses the best FID of StyleGAN2 (at 88 M images) after just 1.1 M images on LSUN-Church. Right: Projected FastGAN yields significantly improved FID scores, even when using subsets of CLEVR with 1k and 10k samples.*

Figure 3.3: Training progress on LSUN church at 256^2 ***pixels.*** *Shown are samples for a fixed noise vector* $\mathbf{z}$ *over k images. From top to bottom: FastGAN, StyleGAN2-ADA, Projected GAN.*

GAN2 after being exposed to only 1.1 M images as compared to 88 M of StyleGAN2. In wall clock time, this corresponds to less than 3 hours instead of 5 days. Hence, from now on, we utilize the FastGAN generator and refer to this model simply as *Projected GAN*.

Fig. 3.3 shows samples for a fixed noise vector $\mathbf{z}$ during training on LSUN-Church. For both FastGAN and StyleGAN, patches of texture gradually morph into a global structure. For Projected GAN, we directly observe the emergence of structure, which becomes more detailed over time. Interestingly, the Projected GAN latent space appears to be very volatile, i.e., for fixed $\mathbf{z}$ the images undergo significant perceptual changes during training. In the non-projected cases, these changes are more gradual. We hypothesize that this induced volatility might be due to the discriminator providing more *semantic* feedback compared to conventional RGB losses. Such semantic feedback could introduce more stochasticity during training which in turn improves convergence and performance. We also observe that the signed real logits of the discriminator remain at the same level throughout training. The signed real logits of the discriminator $sign(D(\mathbf{x}))$ are the portion of the training set that gets positive discriminator outputs. Karras et al. [113] find this a help-

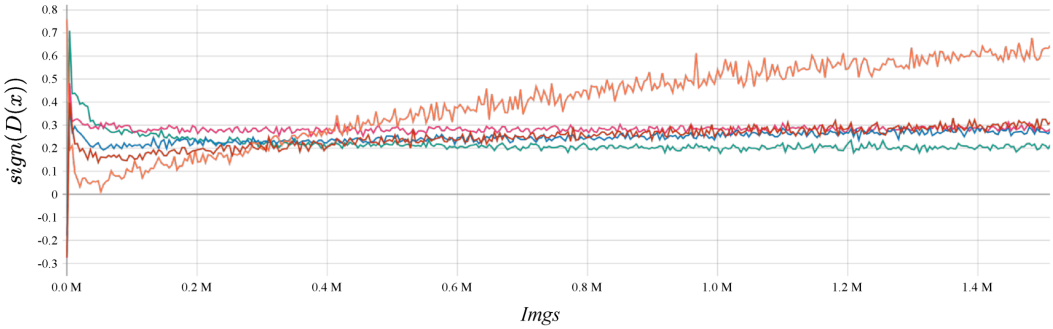

Figure 3.4: Signed Discriminator Logits. *For this experiment, we project through F and train with up to four discriminators; we leave the augmentation probability constant. ($|D_i| = 1$: red, $|D_i| = 2$: blue, $|D_i| = 3$: pink, $|D_i| = 4$: green, RGB baseline: orange). For projected GAN training, the logits remain stable throughout training.*

ful heuristic for quantifying discriminator overfitting. The signed logits should remain constant, which they achieve via adapting the augmentation probability during training. Fig. 3.4 shows that the logits of the RGB baseline steadily increase throughout training, whereas the logits remain mostly constant for Projected GAN training. This observation coincides with the finding that adaptive augmentation is unnecessary for Projected GANs as the logits are already stable.

Sample Efficiency. The use of pretrained models is generally linked to improved sample efficiency. To evaluate this property, we also created two subsets of the 70k CLEVR dataset by randomly sub-sampling 10k and 1k images from it, respectively. As depicted in Fig. 3.2 (right), our Projected GAN significantly improves over both baselines across all dataset splits.

Large Datasets. Besides CLEVR and LSUN-Church, we benchmark Projected GANs against various state-of-the-art models on three other large datasets: LSUN-Bedroom [251] (3M indoor bedroom scenes), FFHQ [115] (70k images of faces) and Cityscapes [45] (25k driving scenes captured from a vehicle). For all datasets, we use an image resolution of 256^2 pixels. As Cityscapes and CLEVR images are not of aspect ratio 1:1 we resize them to 256^2 for training. Besides StyleGAN2-ADA and FastGAN, we compare against SAGAN [256] and GANsformers [100]. All models were trained for 10 M images. For the large datasets, we also report numbers for StyleGAN2 trained for more than 10 M images to report the lowest FID values achieved in previous literature (denoted as StyleGAN2*).

Table 3.3 shows that the Projected GAN outperforms all state-of-the-art models in terms of FID values on all datasets by a large margin. For example, on LSUN-Bedroom, it achieves an FID value of 1.52 compared to 6.15 by GANsformer, the previously best model in this setting. Projected GAN achieves state-of-the-art FID values remarkably fast, e.g., on LSUN-church, it achieves an FID value of 3.18 after 1.1 M *Imgs*. StyleGAN2 has obtained the previously lowest FID value of 3.39 after 88 M *Imgs*, 80 times as many as needed by Projected GAN. Similar speed-ups are also realized for all other large datasets, as shown in Table 3.3. Interestingly, when training longer on FFHQ (39 M *Imgs*), we observe further

improvements of Projected GAN to an FID of 2.2. Note that all five datasets represent very different objects in various scenes. This demonstrates that the performance gain is robust to the choice of the dataset, although the feature network is trained only on ImageNet. It is important to note that the main improvements are based on improved sample diversity, as indicated by recall. The improvement in diversity is most notable on large datasets, e.g., LSUN church, where the image fidelity appears to be similar to StyleGAN.

We also report other metrics that are less benchmarked in GAN literature: KID [19], SwAV-FID [159], precision and recall [196] in Table 3.4 and Table 3.5. These additional metrics mostly reflect the rankings obtained by FID. At high resolutions, Projected GAN performs slightly worse in precision. It appears that Projected GAN incurs small losses in image quality while obtaining better mode coverage. We observe that some samples exhibit artifacts that indicate that Projected GAN training at higher resolutions warrants closer inspection. On small datasets, overfitting is a problem that is not detected well by FID and other metrics [189]. Therefore, it is instructive to inspect latent interpolations for which we refer the reader to the supplementary videos [1]. Projected GAN generates smooth interpolations between random samples on all datasets suggesting that it generalizes rather than memorizing training samples.

Small Datasets. To further evaluate our method in the few-shot setting, we compare against StyleGAN2-ADA and FastGAN on art paintings from WikiArt (1000 images; wikiart.org), Oxford Flowers (1360 images) [166], photographs of landscapes (4319 images; flickr.com), AnimalFace-Dog (389 images) [216] and Pokemon (833 images; pokemon.com). Further, we report results on high-resolution versions of Pokemon and ArtPainting (1024^2). Lastly, we evaluate on AFHQ-Cat, -Dog, and -Wild at 512^2 [40]. The AFHQ datasets contain $\sim$5k closeups per category of cat, dog, or wildlife. We do not have a license to re-distribute these datasets, but we provide the URLs to enable reproducibility, similar to [142].

Projected GAN outperforms all baselines in terms of FID values by a significant margin on all datasets and all resolutions, as shown in Table 3.3. Remarkably, our model beats the prior state-of-the-art on all datasets (256^2) after observing fewer than 600k images. For AnimalFace-Dog, the Projected GAN surpasses the previously best FID after only 20k images. One might argue that the EfficientNet used as feature network facilitates data generation for the animal datasets as EfficientNet is trained on ImageNet, which contains many animal classes (e.g., 120 classes for dog breeds). However, it is interesting to observe that Projected GANs also achieve state-of-the-art FID on Pokemon and Art Painting though these datasets differ significantly from ImageNet. This evidences the generality of ImageNet features. For the high-resolution datasets, Projected GANs achieve the same FID value many times faster than the best baselines, e.g., ten times faster than StyleGAN2-ADA on AFHQ-Cat or four times faster than FastGAN on Pokemon. We remark that F and D_l generalize to any resolution as they are fully convolutional.

[1] https://sites.google.com/view/projected-gan

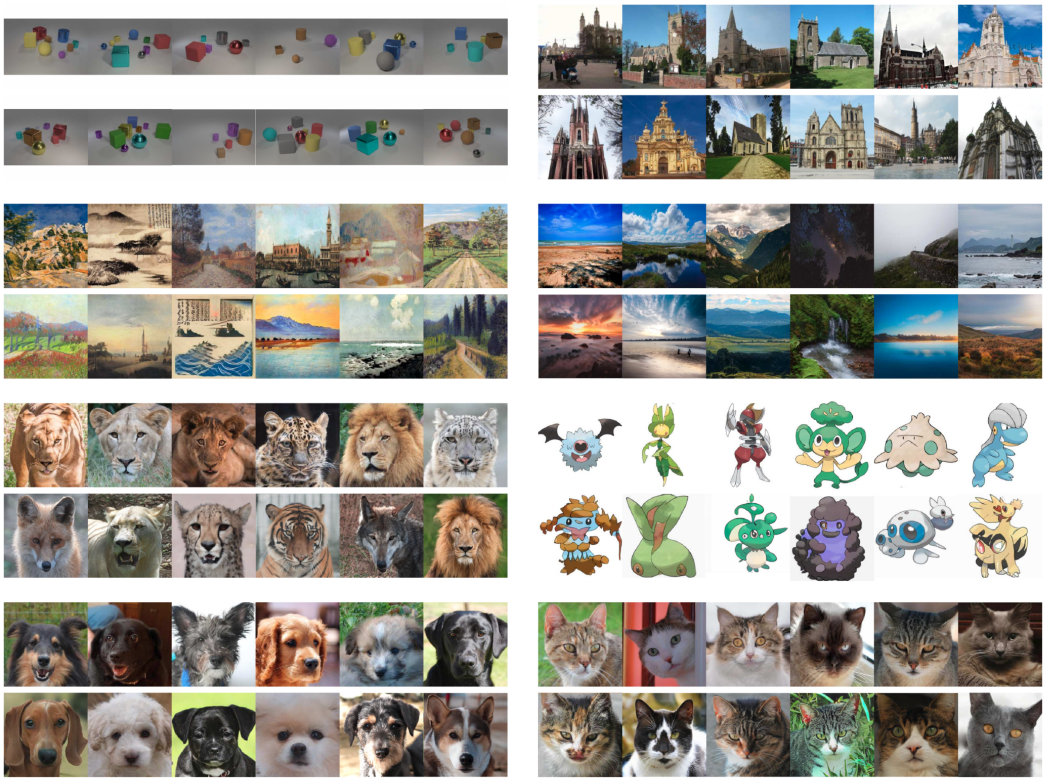

Figure 3.5: Real samples (top rows) vs. samples by Projected GAN (bottom rows). *Datasets (top left to bottom right): CLEVR (256^2), LSUN church (256^2), Art Painting (256^2), Landscapes (256^2), AFHQ-wild (512^2), Pokemon (256^2), AFHQ-dog (512^2), AFHQ-cat (512^2).*

	FID	*Imgs*	FID	*Imgs*	FID	*Imgs*	FID	*Imgs*	FID	*Imgs*
	Large Datasets (256^2)									
	CLEVR		**FFHQ**		**Cityscapes**		**Bedroom**		**Church**	
SAGAN [256]	26.04	10 M	16.21	10 M	12.81	10 M	14.06	10 M	6.15	10 M
StyleGAN2-ADA [113]	10.17	10 M	7.32	10 M	8.35	10 M	11.53	10 M	5.85	10 M
Ganformers [100]	9.24	10 M	7.42	10 M	5.23	10 M	6.15	10 M	5.47	10 M
FastGAN [142]	3.24	10 M	12.69	10 M	8.78	1.8 M	8.24	4.8 M	8.43	8.9 M
Projected GAN	**0.89**	4.5 M	**3.39**	7.1 M	**3.41**	1.7 M	**1.52**	5.2 M	**1.59**	9.2 M
Projected GAN*	3.39	**0.5 M**	3.56	**7.0 M**	4.60	**1.1 M**	2.58	**1.5 M**	3.18	**1.1 M**
StyleGAN2* [254, 113, 115]	5.05	25 M	3.62	25 M	-	-	2.65	70 M	3.39	88 M
	Small Datasets (256^2)									
	Art Painting		**Landscape**		**AnimalFace**		**Flowers**		**Pokemon**	
StyleGAN2-ADA [113]	43.07	3.2 M	15.99	6.3 M	60.90	2.2 M	21.66	3.8 M	40.38	3.4 M
FastGAN [142]	44.02	0.7 M	16.44	1.8 M	62.11	0.2 M	26.23	0.8 M	81.86	2.5 M
Projected GAN	**27.96**	0.8 M	**6.92**	3.5 M	**17.88**	10 M	**13.86**	1.8 M	**26.36**	0.8 M
Projected GAN*	40.22	**0.2 M**	14.99	**0.6 M**	58.07	**0.02 M**	21.60	**0.2 M**	36.57	**0.3 M**
	1024^2				512^2					
	Art Painting		**Pokemon**		**AFHQ-Cat**		**AFHQ-Dog**		**AFHQ-Wild**	
StyleGAN2-ADA [113]	41.69	1.0 M	56.76	0.6 M	3.55	10 M	7.40	10 M	3.05	10 M
FastGAN [142]	46.71	0.8 M	56.46	0.8 M	4.69	1.1 M	13.09	1.6 M	3.14	**1.6 M**
Projected GAN	**32.07**	0.9 M	**33.96**	1.3 M	**2.16**	3.7 M	**4.52**	3.8 M	**2.17**	5.4 M
Projected GAN*	40.33	**0.2 M**	53.74	**0.2 M**	3.53	**1.0 M**	7.10	**0.9 M**	3.03	**1.6 M**

Table 3.3: Quantitative Results. *Projected GAN* reports the point where our approach surpasses the state-of-the-art. StyleGAN2* obtains the lowest FID in previous literature if trained long enough.*

	Large Datasets (256^2)				
	CLEVR	**FFHQ**	**Cityscapes**	**Bedroom**	**Church**
			$FID \downarrow$		
StyleGAN2-ADA [113]	10.17	7.32	8.35	11.53	5.85
FastGAN [142]	3.24	12.69	8.78	8.24	8.43
Projected GAN	**0.89**	**3.08**	**3.41**	**1.52**	**1.59**
			$KID \times 10^3 \downarrow$		
StyleGAN2-ADA [113]	8.15	1.49	3.34	7.42	4.70
FastGAN [142]	2.64	5.34	5.45	5.90	4.61
Projected GAN	**0.51**	**0.44**	**0.91**	**0.36**	**0.50**
			$Precision \uparrow$		
StyleGAN2-ADA [113]	0.373	0.669	**0.649**	0.429	0.565
FastGAN [142]	0.600	**0.716**	0.557	0.602	**0.645**
Projected GAN	**0.640**	0.654	0.619	**0.614**	0.612
			$Recall \uparrow$		
StyleGAN2-ADA [113]	0.569	0.445	0.146	0.202	0.416
FastGAN [142]	0.650	0.184	0.227	0.189	0.207
Projected GAN	**0.735**	**0.464**	**0.361**	**0.346**	**0.438**
			$SwAV-FID \downarrow$		
StyleGAN2-ADA [113]	3.50	1.24	1.35	8.47	2.51
FastGAN [142]	1.46	2.55	1.29	5.38	3.64
Projected GAN	**0.56**	**0.85**	**0.60**	**1.44**	**1.01**
			$CLIP-FID \downarrow$		
StyleGAN2-ADA [113]	4.70	10.3	5.88	42.12	15.85
FastGAN [142]	4.24	19.23	6.46	31.10	35.47
Projected GAN	**0.80**	**7.55**	**2.96**	**11.97**	**13.71**
			$VirTex-FID \downarrow$		
StyleGAN2-ADA [113]	0.78	1.20	1.15	2.20	1.10
FastGAN [142]	0.64	2.47	1.48	2.66	3.61
Projected GAN	**0.35**	**0.64**	**0.49**	**0.81**	**0.82**
			$SWD \times 10^{-3} \downarrow$		
StyleGAN2-ADA [113]	17.50	7.42	10.71	12.53	14.62
FastGAN [142]	28.51	10.19	9.45	14.68	14.42
Projected GAN	**12.90**	**6.41**	**7.27**	**6.83**	**8.37**

Table 3.4: Metrics on Large Datasets *(256^2). Projected GAN compares favorably on most metrics. Exceptions are precision on FFHQ, Cityscapes, and LSUN Church. As argued by [116], shifting from precision to recall is generally desirable, since recall can be traded into precision via truncation.*

	Small Datasets (256^2)				
	Art Painting	**Landscape**	**AnimalFace**	**Flowers**	**Pokemon**
			FID ↓		
StyleGAN2-ADA [113]	43.07	15.99	60.90	21.66	40.38
FastGAN [142]	44.02	16.44	62.11	26.23	81.86
Projected GAN	**27.96**	**6.92**	**17.88**	**13.86**	**26.36**
			KID $\times 10^3$ ↓		
StyleGAN2-ADA [113]	10.23	4.39	22.52	3.56	13.49
FastGAN [142]	13.00	3.40	22.11	6.61	80.30
Projected GAN	**1.25**	**1.30**	**0.03**	**0.38**	**1.32**
			Precision ↑		
StyleGAN2-ADA [113]	0.691	0.709	0.841	0.731	0.735
FastGAN [142]	**0.858**	0.768	0.849	0.611	0.731
Projected GAN	0.762	**0.774**	**0.998**	**0.816**	**0.809**
			Recall ↑		
StyleGAN2-ADA [113]	0.218	0.213	0.036	0.095	0.197
FastGAN [142]	0.044	0.160	0.015	**0.100**	0.004
Projected GAN	**0.239**	**0.258**	**0.095**	0.058	**0.259**
			SwAV $-$ *FID* ↓		
StyleGAN2-ADA [113]	3.32	2.98	16.26	5.02	6.71
FastGAN [142]	3.29	2.42	15.07	7.45	9.25
Projected GAN	**2.25**	**1.42**	**4.22**	**2.70**	**2.04**
			CLIP $-$ *FID* ↓		
StyleGAN2-ADA [113]	44.13	24.89	46.18	26.30	13.96
FastGAN [142]	40.47	19.84	54.69	40.12	87.65
Projected GAN	**22.91**	**13.71**	**16.89**	**15.83**	**9.93**
			VirTex $-$ *FID* ↓		
StyleGAN2-ADA [113]	4.15	2.78	8.83	3.25	3.69
FastGAN [142]	5.72	3.86	9.41	4.08	17.49
Projected GAN	**3.53**	**1.98**	**3.79**	**2.19**	**2.55**
			SWD $\times 10^{-3}$ ↓		
StyleGAN2-ADA [113]	25.55	19.06	22.31	14.04	14.73
FastGAN [142]	21.94	29.87	29.23	17.39	46.81
Projected GAN	**11.44**	**15.38**	**14.34**	**9.61**	**11.65**

Table 3.5: Metrics on Small Datasets *(256^2). Projected GAN performs best on most metrics.*

Figure 3.6: Limitations. *Left: "Floating Heads." Right: Artifacts on FFHQ.*

3.4 Discussion

While we achieve low FID on all datasets, we also identify two systematic failure cases: As depicted in Fig. 3.6, we sometimes observe "floating heads" on AFHQ. In a few samples, the animals appear in high quality but resemble cutouts on blurry or bland backgrounds. We hypothesize that generating a realistic background and image composition is less critical when a prominent object is already depicted. This hypothesis follows from the fact that we used image classification models for the projection, which have been shown to only marginally reduce in accuracy when applied to images of objects with removed background [248]. On FFHQ, Projected GAN sometimes produces poor-quality samples with wrong proportions and artifacts, even at state-of-the-art FID, see Fig. 3.6. This disconnect indicates that features learned on ImageNet, where human faces do not play a significant role, may not be ideal for FFHQ, which is the only dataset where we observe this apparent disconnect between FID and sample quality. An answer could be self-supervised models. We investigate self-supervised features for Projected GAN training in Chapter 4 and Chapter 5.

In terms of generators, StyleGAN is more challenging to tune and does not profit as much from projected training. The FastGAN generator is fast to optimize but simultaneously produced unrealistic samples in some parts of the latent space – a problem that could be solved by a mapping network similar to StyleGAN. Hence, we speculate that unifying the strengths of both architectures in combination with projected training might improve performance further. Combining StyleGAN and Projected GAN will be the focus of the next chapter. Moreover, our study of different pretrained networks indicates that efficient models are especially suitable for Projected GAN training. Exploring this connection in-depth and, in general, determining desirable feature space properties opens up exciting new research opportunities. Lastly, our work advances efficiency for generative models. More efficient models lower the barrier of computational effort needed for generating realistic images. A lower barrier facilitates the malignant use of generative models (e.g., "deep fakes") while simultaneously also democratizing research in this area.

4 Large-Scale Class-Conditional Image-Synthesis with GANs

4.1 Introduction

We showed that Projected GANs are a promising approach to significantly improve training stability, training time, and data efficiency. Interestingly, as we observed, the advantages of Projected GANs only partially extend to StyleGAN on the unimodal datasets we investigated. Style-based GANs (StyleGANs) are a specific instance of GANs, and they exhibit many desirable properties. They achieve high image fidelity [115, 116], fine-grained semantic control [87, 247, 141], and recently alias-free generation enabling realistic animation [114]. Moreover, they reach impressive photorealism on carefully curated datasets, especially of human faces. However, when trained on large and unstructured datasets like ImageNet [49], StyleGANs do not achieve satisfactory results yet. One other problem plaguing generative models, in general, is that they become prohibitively more expensive when scaling to higher resolutions as bigger models are required.

Initially, StyleGAN [115] was proposed to explicitly disentangle factors of variations, allowing for better control and interpolation quality. However, its architecture is more restrictive than a standard generator network [180, 111] which seems to come at a price when training on complex and diverse datasets such as ImageNet. Previous attempts at scaling StyleGAN and StyleGAN2 to ImageNet led to sub-par results [84, 82], giving reason to believe it might be fundamentally limited for highly diverse datasets [84].

BigGAN [24] is the state-of-the-art GAN model for image synthesis on ImageNet. The main factors for BigGANs success are larger batch and model sizes. However, BigGAN has not reached a similar standing as StyleGAN as its performance varies significantly between training runs [113] and as it does not employ an intermediate latent space which is essential for GAN-based image editing [3, 171, 43, 247]. Recently, BigGAN has been superseded in performance by diffusion models [53]. Diffusion models achieve more diverse image synthesis than GANs but are significantly slower during inference and prior work on GAN-based editing is not directly applicable. Following these arguments, successfully training StyleGAN on ImageNet has several advantages over existing methods.

The previously failed attempts at scaling StyleGAN raise the question of whether architectural constraints fundamentally limit style-based generators or if the missing piece is the right training strategy. First, we investigate how to best combine StyleGAN and Projected GANs. We then design a progressive growing strategy tailored to the latest StyleGAN3. These changes, in conjunction with Projected GAN, already allow surpassing prior attempts of training StyleGAN on ImageNet. To further improve results, we analyze the pretrained feature network used for Projected GANs and find that the two standard neu-

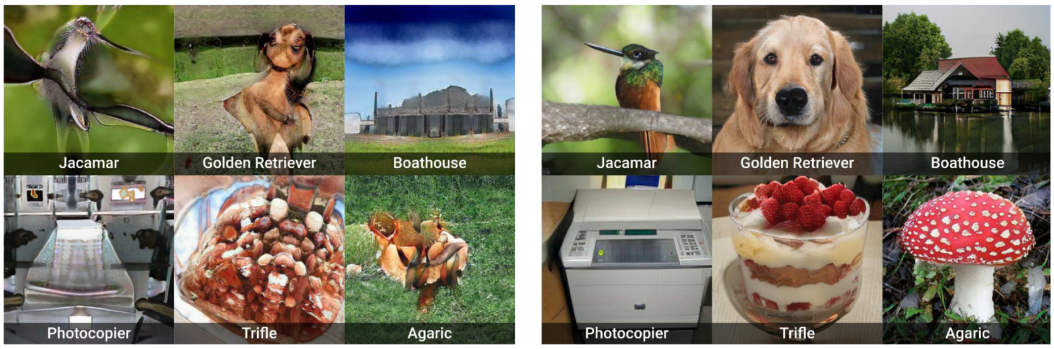

Figure 4.1: *Class-conditional samples generated by StyleGAN3 (left) and StyleGAN-XL (right) trained on ImageNet at resolution* 256^2.

ral architectures for computer vision, CNNs and ViTs [58], significantly improve performance when used jointly. Lastly, we leverage *classifier guidance*, a technique originally introduced for diffusion models to inject additional class-information [53].

Our contributions culminate in a new state-of-the-art on large-scale image synthesis, pushing the performance beyond existing GAN and diffusion models. We showcase inversion and editing for ImageNet classes and find that Pivotal Tuning Inversion (PTI) [190], a powerful new inversion paradigm, combines well with our model and even embeds out-of-domain images smoothly into our learned latent space. Our efficient training strategy allows us to triple the parameters of the standard StyleGAN3 while reaching the prior state-of-the-art performance of diffusion models [53] in a fraction of their training time. It further enables us to be the first to demonstrate image synthesis on ImageNet-scale at a resolution of 1024^2 pixels. We will open-source our code and models upon publication.

4.2 StyleGAN-XL

As mentioned before, StyleGAN has several advantages over existing approaches that work well on ImageNet. But a naïve training strategy does not yield state-of-the-art performance [84, 82]. Our experiments confirm that even the latest StyleGAN3 does not scale well, see Fig. 4.1. Particularly at high resolutions, the training becomes unstable. Therefore, our goal is to train a StyleGAN3 generator on ImageNet successfully. Success is defined in terms of sample quality primarily measured by inception score (IS) [198] and diversity measured by Fréchet Inception Distance (FID) [90]. Throughout this section, we gradually introduce changes to the StyleGAN3 baseline (**Config-A**) and track the improvements in Table 4.1. First, we modify the generator and its regularization losses, adapting the latent space to work well with Projected GAN (**Config-B**) and for the class-conditional setting (**Config-C**). We then revisit progressive growing to improve training speed and performance (**Config-D**). Next, we investigate the feature networks used for Projected GAN training to find a well-suited configuration (**Config-E**). Lastly, we propose classifier guidance for GANs to provide class information via a pretrained classifier (**Config-F**). Our contributions enable us to train a significantly larger model than previously possible while

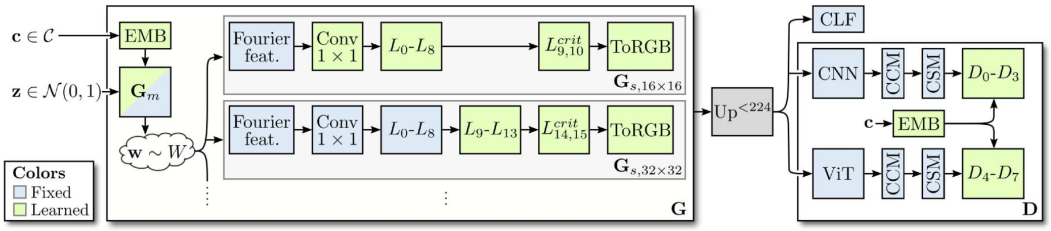

Figure 4.2: Training StyleGAN-XL. *We feed a latent code* $\mathbf{z}$ *and class label* $\mathbf{c}$ *to the pre-trained embedding and the mapping network* $\mathbf{G}_m$ *to generate style codes* $\mathbf{w}$*. The codes modulate the convolutions of the synthesis network* $\mathbf{G}_s$*. During training, we gradually add layers to double the output resolution for each stage of the progressive growing schedule. We only train the latest layers while keeping the others fixed.* $\mathbf{G}_m$ *is only trained for the initial* 16^2 *stage and remains fixed for the higher-resolution stages. The synthesized image is upsampled when smaller than* 224^2 *and passed through a CNN and a ViT and respective feature mixing blocks (CCM+CSM). At higher resolutions, the CNN receives the unaltered image while the ViT receives a downsampled input to keep memory requirements low but still utilize its global feedback. Finally, we apply eight independent discriminators on the resulting multi-scale feature maps. The image is also fed to classifier CLF for classifier guidance.*

requiring less computation than prior art. Our model is three times larger in terms of depth and parameter count than a standard StyleGAN3. However, to match the prior state-of-the-art performance of ADM [53] at a resolution of 512^2 pixels, training the models on a single NVIDIA Tesla V100 takes 400 days compared to the previously required 1914 V100-days. We refer to our model as **StyleGAN-XL** (Fig. 4.2).

4.2.1 Adapting Regularization and Architectures

Training on a diverse and class-conditional dataset makes it necessary to introduce several adjustments to the standard StyleGAN configuration. We construct our generator architecture using layers of StyleGAN3-T, the translational-equivariant configuration of Style-

	Configuration	FID ↓	IS ↑
A	StyleGAN3	53.57	15.30
B	+ Projected GAN & small $\mathbf{z}$	22.98	57.62
C	+ Pretrained embeddings	20.91	35.79
D	+ Progressive growing	19.51	35.74
E	+ ViT & CNN as $\mathbf{F}_{1,2}$	12.43	56.72
F	+ CLF guidance (StyleGAN-XL)	**12.24**	**86.21**

Table 4.1: Ablation Study on ImageNet 128^2*. Results for different configurations after training for* 15 *V100-days.*

GAN3. In initial experiments, we found the rotational-equivariant StyleGAN3-R to generate overly symmetric images on more complex datasets, resulting in kaleidoscope-like patterns.

Regularization. In GAN training, it is common to use regularization for both the generator and the discriminator. Regularization improves results on uni-modal datasets like FFHQ [115] or LSUN [251], whereas it can be detrimental on multi-modal datasets [24, 84]. Therefore, we aim to avoid regularization when possible. [114] find style mixing to be unnecessary for the latest StyleGAN3; hence, we also disable it. Path length regularization can lead to poor results on complex datasets [84] and is, per default, disabled for StyleGAN3 [114]. However, path length regularization is attractive as it enables high-quality inversion [116]. We also observe unstable behavior and divergence when using path length regularization in practice. We found that this problem can be circumvented by only applying regularization after the model has been sufficiently trained, i.e., after 200k images. For the discriminator, we use spectral normalization without gradient penalties. In addition, we blur all images with a Gaussian filter with $\sigma = 2$ pixels for the first $200k$ images. Discriminator blurring has been introduced in [114] for StyleGAN3-R. It prevents the discriminator from focusing on high frequencies early on, which we found beneficial across all settings we investigated.

Low-Dimensional Latent Space. As we observed in Chapter 3, Projected GANs work better with FastGAN [142] than with StyleGAN. One main difference between these generators is their latent space; StyleGAN's latent space is comparatively high dimensional (FastGAN: $\mathbb{R}^{100}$, BigGAN: $\mathbb{R}^{128}$, StyleGAN: $\mathbb{R}^{512}$). Recent findings indicate that the *intrinsic dimension* of natural image datasets is relatively low [177], ImageNet's dimension estimate is around 40. Accordingly, a latent code of size 512 is highly redundant, making the mapping network's task harder at the beginning of training. Consequently, the generator is slow to adapt and cannot benefit from Projected GAN's speed up. We, therefore, reduce StyleGAN's latent code $\mathbf{z}$ to 64 and now observe stable training in combination with Projected GAN, resulting in lower FID than the baseline (**Config-B**). We keep the original dimension of the *style code* $\mathbf{w} \in \mathbb{R}^{512}$ to not restrict the model capacity of the mapping network $\mathbf{G}_m$.

Pretrained Class Embeddings. Conditioning the model on class information is essential to control the sample class and improve overall performance. A class-conditional variant of StyleGAN was first proposed in [113] for CIFAR10 [125] where a one-hot encoded label is embedded into a 512-dimensional vector and concatenated with $\mathbf{z}$. For the discriminator, class information is projected onto the last discriminator layer [154]. We observe that **Config-B** tends to generate similar samples per class resulting in high IS. To quantify mode coverage, we leverage the recall metric [129] and find that **Config-B** achieves a low recall of 0.004. We hypothesize that the class embeddings collapse when training with Projected GAN. Therefore, to prevent this collapse, we aim to ease the optimization of the embeddings via pretraining. We extract and spatially pool the lowest resolution features of an Efficientnet-lite0 [225] and calculate the mean per ImageNet class. The network has a low channel count to keep the embedding dimension small, following the arguments of the

previous section. The embedding passes through a linear projection to match the size of $\mathbf{z}$ to avoid an imbalance. Both $\mathbf{G}_m$ and $\mathbf{D}_i$ are conditioned on the embedding. During GAN training, the embedding and the linear projection are optimized to allow specialization. Using this configuration, we observe that the model generates diverse samples per class, and recall increases to 0.15 (**Config-C**). Note that for all configurations in this ablation, we restrict the training time to 15 V-100 *days*. Hence, the absolute recall is markedly lower compared to the fully trained models. Conditioning a GAN on pretrained features was also recently investigated by [29]. In contrast to our approach, [29] condition on specific *instances*, instead of learning a general class embedding.

4.2.2 Reintroducing Progressive Growing

Progressively growing the output resolution of a GAN was introduced by [111] for fast and more stable training. The original formulation adds layers during training to both **G** and **D** and gradually fades in their contribution. However, in a later work, it was discarded [116] as it can contribute to texture-sticking artifacts. Recent work by [114] finds that the primary cause of these artifacts is aliasing, so they redesign each layer of StyleGAN to prevent it. This motivates us to reconsider progressive growing with a carefully crafted strategy that aims to suppress aliasing as best as possible. Training first on very low resolutions, as small as 16^2 pixels, enables us to break down the daunting task of training on high-resolution ImageNet into smaller subtasks. This idea is in line with the latest work on diffusion models [165, 195, 53, 92]. They observe considerable improvements in FID on ImageNet when using a two-stage model, i.e., stacking an independent low-resolution model and an upsampling model to generate the final image.

Commonly, GANs follow a rigid sampling rate progression, i.e., at each resolution, there is a fixed amount of layers followed by an upsampling operation using fixed filter parameters. StyleGAN3 does not follow such a progression. Instead, the layer count is set to 14, independent of the output resolution, and the filter parameters of up- and downsampling operations are carefully designed for antialiasing under the given configuration. The last two layers are critically sampled to generate high-frequency details. When adding layers for the subsequent highest resolution, discarding the previously critically sampled layers is crucial as they would introduce aliasing when used as intermediate layers [116, 114]. Furthermore, we adjust the filter parameters of the added layers to adhere to the flexible layer specification of [114]. We start progressive growing at resolution 16^2 using 11 layers. The layer specifications are computed according to [114] and remain fixed for the remaining training. For the next stage, at resolution 32^2, we discard the last 2 layers and add 7 new ones. The specifications for the new layers are computed according to [114] for a model with resolution 32^2 and 16 layers. Continuing this strategy up to resolution 1024^2 yields the flexible layer specification of StyleGAN-XL in Fig. 4.3. In contrast to [111] we do not add layers to the discriminator. Instead, to fully utilize the pretrained feature network **F**, we upsample both data and synthesized images to **F**'s training resolution (224^2 pixels) when training on smaller images.

We start progressive growing at a resolution of 16^2 using 11 layers. Every time the resolution increases, we cut off 2 layers and add 7 new ones. Empirically, fewer layers

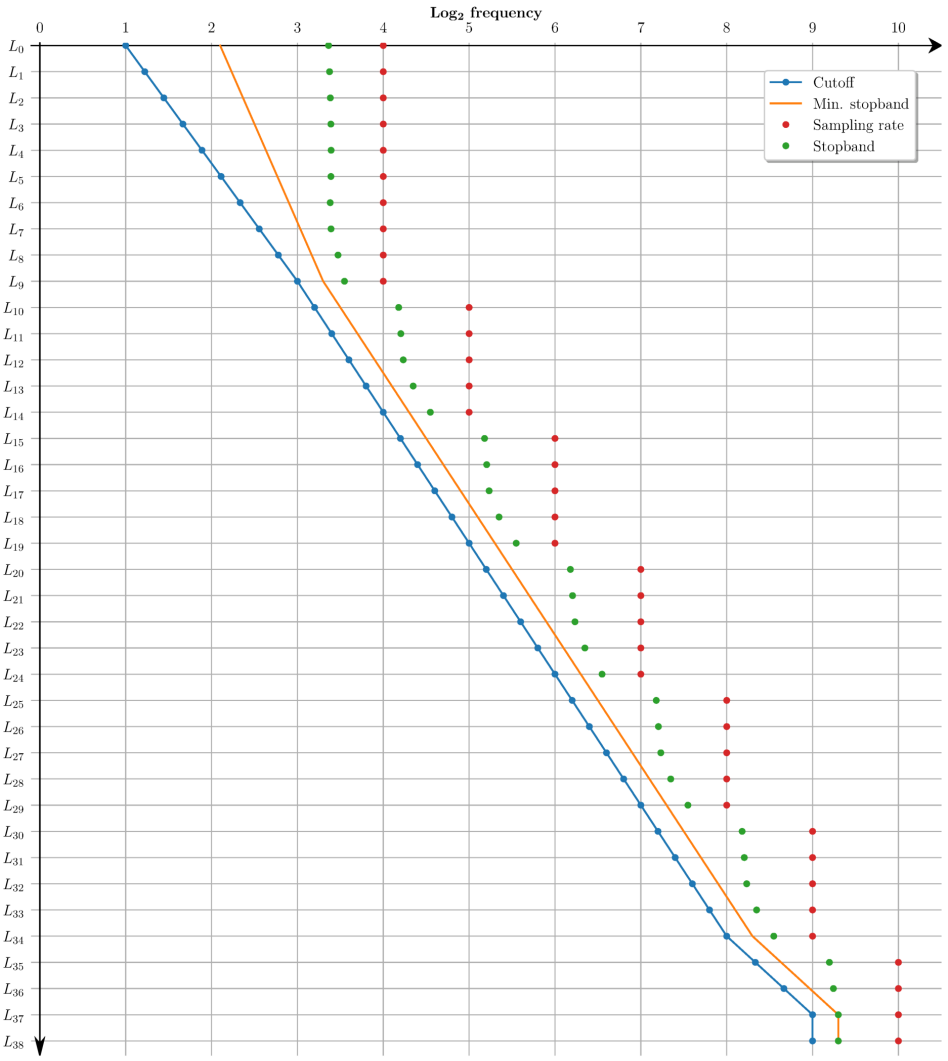

Figure 4.3: Flexible Layer Specification of Stylegan-XL. *StyleGAN-XL consists of* 39 *layers at resolution* 1024^2*. Cutoff (blue) and minimum acceptable stopband frequency (orange) obey geometric progression over the layers; sampling rate (red) and actual stopband (green) are computed according to our design constraints.*

result in worse performance; adding more leads to increased overhead and diminishing returns. For the final stage at 1024^2, we add only 5 layers as the last two are not discarded. This amounts to 39 layers at the maximum resolution of 1024^2. Instead of a fixed growing schedule, each stage is trained until FID stops decreasing. We find it beneficial to use a large batch size of 2048 on lower resolution (16^2 and 32^2), similar to [24]. On higher resolutions, smaller batch sizes suffice (64^2 to 256^2: 256, 512^2 to 1024^2: 128). Once new layers are added, the lower-resolution layers remain fixed to prevent mode collapse.

In our ablation study, FID improves only slightly (**Config-D**) compared to **Config-C**. However, the main advantage can be seen at high resolutions, where progressive growing drastically reduces training time. At resolution 512^2, we reach the prior state-of-the-art (FID = 3.85) after 2 V100-days. This reduction is in contrast to other methods such as ADM, where doubling the resolution from 256^2 to 512^2 pixels corresponds to increasing training time from 393 to 1914 V100-days to find the best-performing model[1]. As our aim is not to introduce texture sticking artifacts, we measure $EQ\text{-}T$, a metric for determining translation equivariance [114], where higher is better. **Config-C** yields $EQ\text{-}T = 55$, while **Config-D** attains $EQ\text{-}T = 48$. This only slight reduction in equivariance shows that **Config-D** restricts aliasing almost as well as a configuration without growing. For context, architectures with aliasing yield $EQ\text{-}T \sim 15$.

4.2.3 Exploiting Multiple Feature Networks

The ablation study conducted in 3 finds that most pretrained feature networks **F** perform similarly in terms of FID when used for Projected GAN training regardless of training data, pretraining objective, or network architecture. However, the study does not answer if combining several **F** is advantageous. Starting from the standard configuration, an EfficientNet-lite0, we add a second **F** to inspect the influence of its pretraining objective (classification or self-supervision) and architecture (CNN or Vision Transformer (ViT) [58]). The results in Table 4.2 show that an additional CNN leads to slightly lower FID. Combining networks with different pretraining objectives does not offer benefits over using two classifier networks. However, combining an EfficientNet with a ViT improves performance significantly. This result corroborates recent results in neural architecture literature, which find that supervised and self-supervised representations are similar [81], whereas ViTs and CNNs learn different representations [182]. Combining both architectures appears to have complementary effects for Projected GANs. We do not see significant improvements when adding more networks; hence, **Config-E** uses the combination of EfficientNet [225] and DeiT-base [230].

4.2.4 Classifier Guidance for GANs

Dhariwal and Nichol [53] introduced classifier guidance to inject class information into diffusion models. Classifier guidance modifies each diffusion step at time step t by adding

[1] Note that these settings are not directly comparable as the stem of our model is pretrained, but the values should give a general sense of the order of magnitude.

Model		Type		Objective		FID ↓	IS ↑
$\mathbf{F}_1$	$\mathbf{F}_2$	$\mathbf{F}_1$	$\mathbf{F}_2$	$\mathbf{F}_1$	$\mathbf{F}_2$		
EffNet		CNN		Class		19.51	35.74
EffNet	ResNet50	CNN	CNN	Class	Class	16.16	49.13
EffNet	ResNet50	CNN	CNN	Class	Self	18.53	38.26
EffNet	DeiT-M	CNN	ViT	Class	Class	**12.43**	**56.72**

Table 4.2: Feature Network Study on ImageNet 128^2. *Comparing combinations of different feature networks* **F**. *Beginning from the base configuration using an EfficientNet-lite0 (EffNet), we add a second* **F** *with varying architecture type and pretraining objective (Class: Classification, Self: MoCo-v2 [38]).*

gradients of a pretrained classifier $\nabla_{\mathbf{x}_t} \log p_\phi(\mathbf{c}|x_t,t)$. The best results are obtained by applying guidance on class-conditional models and scaling the classifier gradients by a constant $\lambda > 1$. This combination indicates that our model may also profit from classifier guidance, even though it already receives class information via embeddings.

We first pass the generated image $\mathbf{x}$ through a pretrained classifier CLF to predict the class label c_i. We then add a cross-entropy loss $\mathcal{L}_{CE} = -\sum_{i=0}^{C} c_i \log CLF(x_i)$ as an additional term to the generator loss and scale this term by a constant λ. For the classifier, we use DeiT-small [230], which exhibits strong classification performance while not adding much overhead to the training. Similar to [53], we observe a significant improvement in IS, indicating an increase in sample quality (**Config-F**). We find $\lambda = 8$ to work well empirically. Classifier guidance only works well on higher resolutions ($> 32^2$); otherwise, it leads to mode collapse. This is in contrast to [53], who exclusively guide their low-resolution model. The difference stems from how guidance is applied: we use it for model training, whereas [53] guides the sampling process.

4.3 Experiments

In this section, we first compare StyleGAN-XL to the state-of-the-art approaches for image synthesis on ImageNet. We then evaluate the inversion and editing capabilities of StyleGAN-XL. As described above, we scale our model to a resolution of 1024^2 pixels, which no prior work has attempted so far on ImageNet. An initial challenge is the lack of high-resolution data; the mean resolution of ImageNet is 469×387. Similar to the procedure used for generating CelebA-HQ[111], we preprocess the whole dataset with SwinIR-Large [136], a recent model for real-world image super-resolution. Of course, a trivial way of achieving good performance on this dataset would be to draw samples from a 256^2 generative model and pass it through SwinIR. However, SwinIR adds significant computational overhead as it is 60 times slower than our upsampling stack. Furthermore, this way, StyleGAN-XL's weights can be used for initialization when finetuning on other high-resolution datasets. Lastly, combining StyleGAN-XL and SwinIR would impair translation

equivariance.

4.3.1 Image Synthesis

Both our work and [53] use classifier networks to guide the generator. To ensure the models are not inadvertently optimizing for FID and IS, which also utilize a classifier network, we propose random-FID (rFID). For rFID, we calculate the Fréchet distance in the `pool_3` layer of a randomly initialized inception network [224]. The efficacy of random features for evaluating generative models has been demonstrated in [160]. Furthermore, we report sFID [161] to assess spatial structure. Lastly, sample fidelity and diversity are evaluated via precision and recall [129].

In Table 4.3, we compare StyleGAN-XL to the currently strongest GAN model (BigGAN-deep [24]) and diffusion models (CDM [92], ADM [53]) on ImageNet. The values for ADM are calculated with and without additional methods (Upsampling **U** and Classifier Guidance **G**). For StyleGAN2, we report numbers by [82]. We find that StyleGAN-XL substantially outperforms all baselines across all resolutions in FID, sFID, rFID, and IS. An exception is recall, according to which StyleGAN-XL's sample diversity lies between BigGAN and ADM, making progress in closing the gap between these model types. BigGAN's sample quality is the best among all compared approaches, which comes at the price of significantly lower recall. StyleGAN-XL allows for the truncation trick to increase sample fidelity, i.e., we can interpolate a sampled style code w with the class-wise mean style code $\bar{w}$. We observe that for StyleGAN-XL, truncation does not increase precision, indicating that developing novel truncation methods for high-diversity GANs is an exciting research direction for future work. Interestingly, StyleGAN-XL attains high diversity across all resolutions, which can be attributed to our progressive growing strategy. Furthermore, this strategy enables to scale to megapixel resolution successfully. Training at 1024^2 for a single V100-day yields a noteworthy FID of 2.8. At this resolution, we do not compare to baselines because of resource constraints, as they are prohibitively expensive to train. visualizes generated samples at increasing resolutions. Fig. 4.4 visualizes generated samples at increasing resolutions. Fig. 4.5 shows additional interpolations between samples from different classes. Fig. 4.6. Moreover, we train an unconditional variant of StyleGAN-XL without class-embeddings and classifier guidance, Fig. 4.7 show samples on FFHQ 1024^2 and Pokemon 1024^2 generated by this model. Lastly, we compare BigGAN, ADM, and StyleGAN-XL on different ImageNet classes in Fig. 4.8, Fig. 4.9, and Fig. 4.10. For a fair comparison, we do not use truncation or classifier guidance. Instead, we show images with the largest logits given by a VGG16, which corresponds to individual image quality.

4.3.2 Inversion and Manipulation

GAN-editing methods first *invert* a given image into latent space, i.e., find a style code w that reconstructs the image as faithful as possible when passed through $\mathbf{G}_s$. Then, w can be manipulated to achieve semantically meaningful edits [76, 213].

Model	FID ↓	sFID ↓	rFID ↓	IS ↑	Pr ↑	Rec ↑	Model	FID ↓	sFID ↓	rFID ↓	IS ↑	Pr ↑	Rec ↑
Resolution 128^2							**Resolution** 256^2						
BigGAN	6.02	7.18	6.09	145.83	**0.86**	0.35	StyleGAN2	49.20					
CDM	3.52	128.80		128.80			BigGAN	6.95	7.36	75.24	202.65	**0.87**	0.28
ADM	5.91	5.09	13.29	93.31	0.70	**0.65**	CDM	4.88	158.70		158.70		
ADM-G	2.97	5.09	3.80	141.37	0.78	0.59	ADM	10.94	6.02	125.78	100.98	0.69	**0.63**
StyleGAN-XL	**1.81**	**3.82**	**1.82**	**200.55**	0.77	0.55	ADM-G-U	3.94	6.14	11.86	215.84	0.83	0.53
							StyleGAN-XL	**2.30**	**4.02**	**7.06**	**265.12**	0.78	0.53
Resolution 512^2							**Resolution** 1024^2						
BigGAN	8.43	8.13	312.00	177.90	**0.88**	0.29	StyleGAN-XL	**2.52**	**4.12**	**413.12**	**260.14**	**0.76**	**0.51**
ADM	23.24	10.19	561.32	58.06	0.73	**0.60**							
ADM-G-U	3.85	5.86	210.83	221.72	0.84	0.53							
StyleGAN-XL	**2.41**	**4.06**	**51.54**	**267.75**	0.77	0.52							

***Table 4.3: Image Synthesis on ImageNet**. Empty cells indicate that the model was not available and the respective metric was not evaluated in the original work.*

***Figure 4.4: Samples at Different Resolutions Using the Same* w.** *The samples are generated by the models obtained during progressive growing. We upsample all images to* 1024^2 *using nearest-neighbor interpolation for visualization purposes. Zooming in is recommended.*

Figure 4.5: Interpolations. *StyleGAN-XL generates smooth interpolations between samples of different classes.*

Figure 4.6: Samples on FFHQ 1024^2.

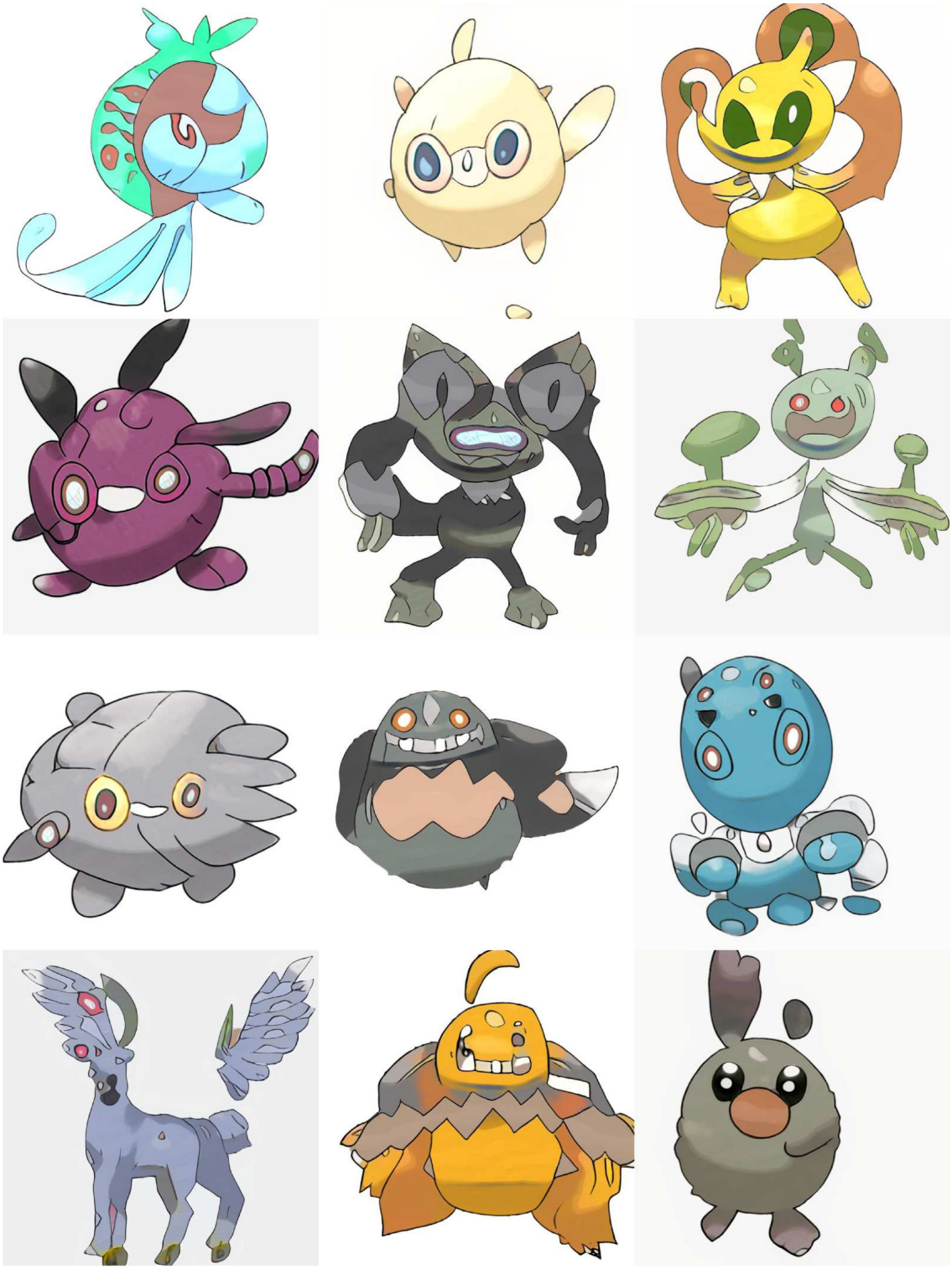

Figure 4.7: Samples on Pokemon 1024^2.

Figure 4.8: Qualitiative Comparison on ImageNet 256^2*.. We compare BigGAN (left column), ADM (middle column), and StyleGAN-XL (right column). Classes from top to bottom: pizza, valley, daisy, dough, comic book.*

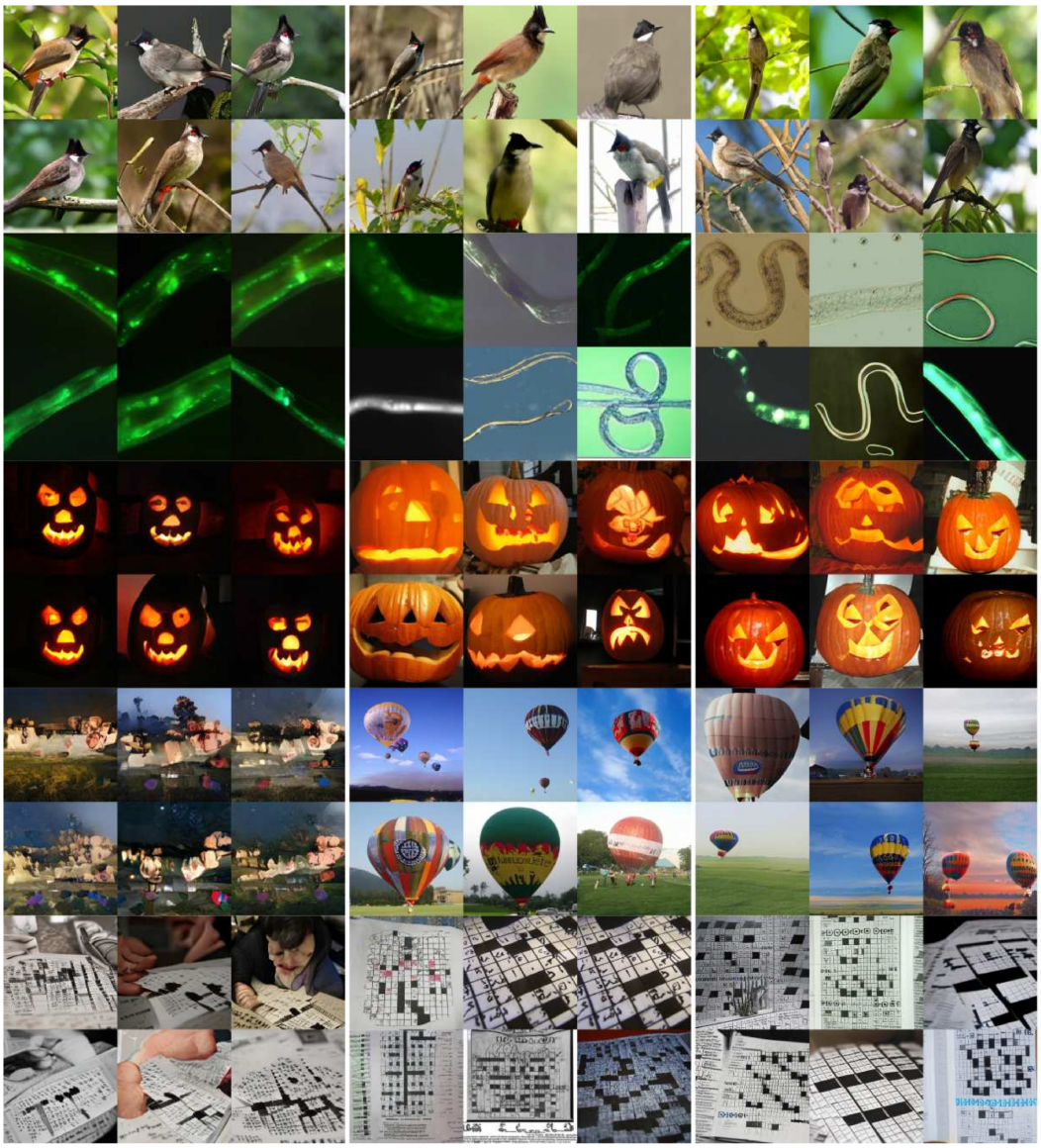

Figure 4.9: Qualitiative Comparison on ImageNet 256^2*.. We compare BigGAN (left column), ADM (middle column), and StyleGAN-XL (right column). Classes from top to bottom: bulbul, nematode, jack-o'-lantern, balloon, crossword puzzle.*

Figure 4.10: Qualitiative Comparison on ImageNet 256^2**.**. *We compare BigGAN (left column), ADM (middle column), and StyleGAN-XL (right column). Classes from top to bottom: agaric, orange, Tibetian mastiff, espresso, paddlewheel.*

Inversion. Standard approaches for inverting $\mathbf{G}_s$ use either latent optimization [1, 46, 116] or an encoder [173, 5, 233]. A common way to achieve low reconstruction error is to use an extended definition of the latent space: $\mathcal{W}+$. For $\mathcal{W}+$ a separate $\mathbf{w}$ is chosen for each layer of $\mathbf{G}_s$. However, as highlighted by [268, 233], this extended definition achieves higher reconstruction quality in exchange for lower editability. Therefore, [233] carefully design an encoder to maintain editability by mapping to regions of $\mathcal{W}+$ that are close to the original distribution of $\mathcal{W}$. Following [116], we use basic latent optimization in $\mathcal{W}$

Model	MSE ↓	PSNR ↑	SSIM ↑	FID ↓
BigGAN	0.10	10.85	0.26	47.48
StyleGAN-XL	**0.06**	**13.45**	**0.33**	**21.73**

Table 4.4: Inversion Results. *The metrics are computed between the inversions obtained by the model and the reconstruction targets.*

for inversion. Given a target image, we first compute its average style code $\bar{\mathbf{w}}$ by running 10000 random latent codes $\mathbf{z}$ and target specific class samples $\mathbf{c}$ through the mapping network. As the class label of the target image is unknown, we pass it to a pretrained classifier. We then use the classifier logits as a multinomial distribution to sample $\mathbf{c}$. In our experiments, we use Deit-base [230] as a classifier, but other choices are possible. At the beginning of optimization, we initialize $\mathbf{w} = \bar{\mathbf{w}}$. The components of $\mathbf{w}$ are the only trainable parameters. The optimization runs for 1000 iterations using the Adam optimizer [118] with default parameters. We optimize the LPIPS [259] distance between the target image and the generated image. For StyleGAN-XL, the maximum learning rate is $\lambda_{max} = 0.05$. It is ramped up from zero linearly during the first 50 iterations and ramped down to zero using a cosine schedule during the last 250 iterations. For BigGAN, we empirically found $\lambda_{max} = 0.001$ and a ramp-down over the last 750 iterations to yield the best results. All inversion experiments are performed at resolution 512^2 and computed on $5k$ images (10% of the validation set). We report the results in Table 4.4 and show qualitative results in Fig. 4.11. We find that StyleGAN-XL already achieves satisfactory inversion results using basic latent optimization. For inversion on the ImageNet validation set at 512^2, StyleGAN-XL yields PSNR $= 13.5$ on average, improving over BigGAN at PSNR $= 10.8$. Besides better pixel-wise reconstruction, StyleGAN-XL's inversions are semantically closer to the target images. We measure the FID between reconstructions and targets, and StyleGAN-XL attains FID $= 21.7$ while BigGAN reaches FID $= 47.5$.

Given the results above, it is also possible to further refine the obtained reconstructions. [190] recently introduced pivotal tuning inversion (PTI). PTI uses an initial inverted style code as a pivot point around which the generator is finetuned. Additional regularization prevents altering the generator output far from the pivot. Combining PTI with StyleGAN-XL allows us to invert both in-domain (ImageNet validation set) and out-of-domain images almost precisely. At the same time, the generator output remains perceptually smooth, see Fig. 4.12.

Image Manipulation. Given the inverted images, we can leverage GAN-based editing methods [237, 87, 214, 120, 222] to manipulate the style code $\mathbf{w}$. In Fig. 4.13 (Left), we first invert a given source image via latent space optimization. We can then apply manipulation directions obtained by, e.g., GANspace [87]. Prior work [104] also investigates in-plane translation. This operation can be directly defined in the input grid of StyleGAN-XL. The input grid also allows performing extrapolation; see Fig. 4.13 (Left).

An inherent property of StyleGAN is the ability of style mixing by supplying the style

Figure 4.11: Inversion of a Given Source Image. *For BigGAN, we invert to its latent space* **z**, *for StyleGAN-XL we invert to style codes* **w**.

Figure 4.12: Interpolations. *StyleGAN-XL generates smooth interpolations between samples of different classes (Row 1 & Row 2). PTI allows inverting to the latent space with low distortion (outermost image, Row 3 & Row 4), and consistently embeds out-of-domain inputs, such as the one on the bottom right.*

codes of two samples to different layers of $\mathbf{G}_s$, generating a hybrid image. This hybrid takes on different semantic properties of both inputs. Style mixing is commonly employed for instances of a single domain, i.e., combining two human portraits. StyleGAN-XL inherits this ability and, to a certain extent, even generates out-of-domain combinations between different classes, akin to counterfactual images [204]. This technique works best for aligned samples, similar to StyleGAN's originally favored setting, FFHQ. Curated examples are shown in Fig. 4.13 (Right).

4.4 Discussion

Our contributions allow StyleGAN to accomplish state-of-the-art high-resolution image synthesis on ImageNet. Furthermore, applying it to big and small unimodal datasets is straightforward, and we also achieve state-of-the-art performance on FFHQ and Pokemon at resolution 1024^2. Exploring new editing methods and dataset generation [30, 134] using StyleGAN-XL are exciting future avenues. Furthermore, future work may tackle an even larger megapixel dataset. However, a larger yet diverse dataset is not available so far. Current large-scale, high-resolution datasets are of single object classes or contain many similar images [261, 70, 174].

Several limitations of the current model remain, which we discuss below.

Architectural Limitations. First, StyleGAN-XL is three times larger than StyleGAN3, constituting a higher computational overhead when used as a starting point for finetuning. Therefore, it will be worth exploring GAN distillation methods [32] that trade-off performance for model size. Second, StyleGAN-XL uses translation-equivariant layers of StyleGAN3-T. As described above, StyleGAN3-R tends to produce overly symmetrical images and adds significant computational overhead. Finding a more efficient rotational-equivariant architecture is an important future direction. Second, we find StyleGAN3, and consequently, StyleGAN-XL, harder to edit, e.g., high-quality edits via $\mathcal{W}$ are noticeably easier to achieve with StyleGAN2. As already observed in [114], StyleGAN3's semantic controllability is reduced for the sake of equivariance. However, techniques using the *StyleSpace* [247], e.g., StyleMC [120], tend to yield better results in our experiments, confirming the findings of concurrent work by [6]. Furthermore, we remark that our framework can also easily be used with StyleGAN2 layers.

Comparison to Diffusion Models. Our model is larger than earlier StyleGANs, yet it is still several orders of magnitudes faster than ADM; we compare inference speeds in Table 4.5. Low data coverage is a known problem of GANs, and StyleGAN-XL makes notable headway on this issue. However, StyleGAN-XL is still outperformed by diffusion models regarding data coverage. Furthermore, classes of unaligned humans and human faces are particularly challenging for all compared approaches, likely due to ImageNet's emphasis on non-human objects [53]. For such classes, we observe that ADM [53] generates more convincing human faces than BigGAN [24] or StyleGAN-XL. Both GANs can synthesize realistic faces; however, the main challenge in this setting is that the dataset is

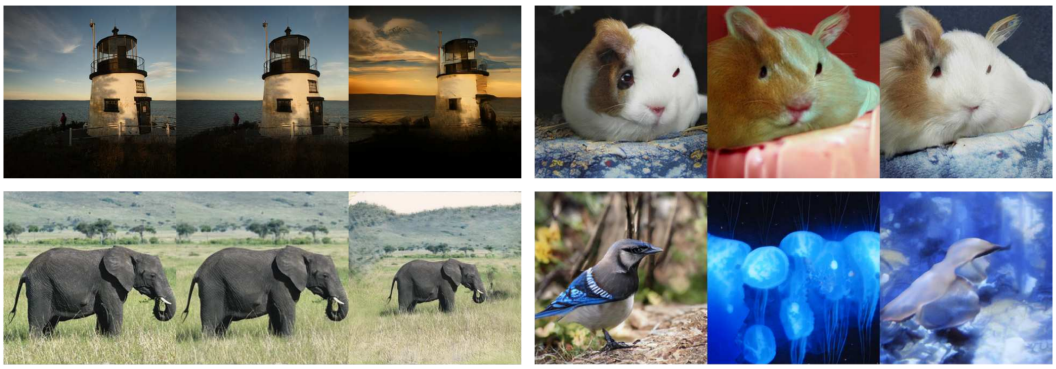

Figure 4.13: Image Editing and Style Mixing. *Left: First, a given image is inverted via PTI [190]. Right: Given two images, we can mix their styles. This methods works for samples of the same or similar classes, and to a certain extent, for distant classes. For this experiment, we utilize random samples instead of inversions.*

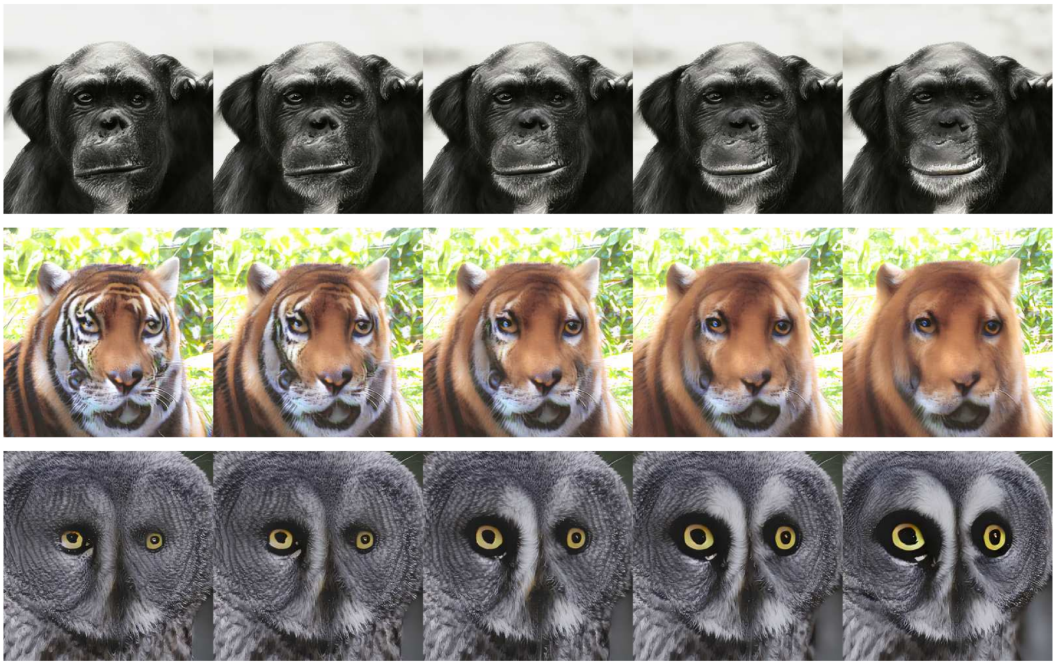

Figure 4.14: Image Manipulation via Language. *Given a random sample, we manipulate the image by by following semantic directions in latent space found by StyleMC [120]. The latent space directions from top to bottom are: "smile", "no stripes", and "big eyes".*

ADM BigGAN StyleGAN-XL

Figure 4.15: Imagenet Classes Containing Humans. *Samples for BigGAN and ADM are taken from [53].*

Model	**Inference Time** ↓		
	Res. 128^2	**Res.** 256^2	**Res.** 512^2
ADM	27.07	40.26	91.54
StyleGAN-XL	0.05	0.07	0.10

Table 4.5: Inference speed comparison.. *We measure the time required for a forward pass with batch size* 1 *in V100-seconds. ADM uses classifier guidance.*

unstructured, and the humans are not aligned. [24] remarked on the particular challenge of classes containing details to which human observers are more sensitive. We show examples in Fig. 4.15. Whether the points above are a general limitation of GANs remains an interesting open question for future research.

5 Large-Scale Text-to-Image Synthesis with GANs

5.1 Introduction

We have shown that GANs can be scaled beyond datasets of medium-sized, uni-modal datasets. A natural progression from this point is to further upscale — from one million images contained in ImageNet to several hundreds of millions of data points. At this scale, generative modeling commonly becomes text-conditional. In text-to-image synthesis, novel images are generated based on text prompts. The state-of-the-art in this task has recently taken dramatic leaps forward thanks to two key ideas. First, using a large pre-trained language model as an encoder for the prompts makes it possible to condition the synthesis based on general language understanding [183, 194]. Second, using large-scale training data consisting of hundreds of millions of image-caption pairs [208] allows the models to synthesize almost anything imaginable.

Training datasets continue to increase rapidly in size and coverage. Consequently, text-to-image models must be scalable to a large capacity to absorb the training data. Recent successes in large-scale text-to-image generation have been driven by diffusion models (DM) [183, 194, 191] and autoregressive models (ARM) [257, 253, 72] that seem to have this property built in, along with the ability to deal with highly multi-modal data.

Interestingly, GANs — the dominant family of generative models in smaller and less diverse datasets — have not been particularly successful in this task [267]. Our goal is to show that they can regain competitiveness.

The primary benefits offered by GANs are inference speed and control of the synthesized result via latent space manipulations. StyleGAN [115, 116, 114] in particular has a thoroughly studied latent space, which allows principled control of generated images [18, 87, 212, 2, 109]. While there has been notable progress in speeding up DMs [199, 112, 146], they are still far behind GANs that require only a single forward pass.

We draw motivation from the observation that GANs lagged similarly behind diffusion models in ImageNet [48, 53] synthesis until we redesigned the discriminator architecture for StyleGAN-XL, which allowed GANs to close the gap. In Section 5.2, we start from StyleGAN-XL and revisit the generator and discriminator architectures, considering the requirements specific to the large-scale text-to-image task: large capacity, extremely diverse datasets, strong text alignment, and controllable variation vs. text alignment tradeoff.

We have a fixed training budget of 4 weeks on 64 NVIDIA A100s available for training our final model at scale. This constraint forces us to set priorities because the budget is likely insufficient for state-of-the-art, high-resolution results [44]. While the ability of GANs to scale to high resolutions is well known [241, 116], successful scaling to the large-

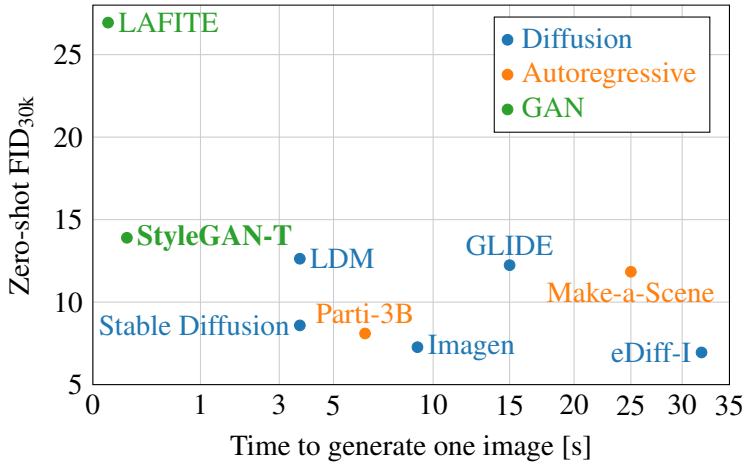

Figure 5.1: Quality vs. speed *in large-scale text-to-image synthesis. StyleGAN-T greatly narrows the quality gap between GANs and other model families while generating samples at a rate of 10 FPS on an NVIDIA A100. The y-axis corresponds to zero-shot FID on MS COCO at 256×256 resolution; lower is better.*

scale text-to-image task remains undocumented. We thus focus primarily on solving this task in lower resolutions, dedicating only a limited budget to the super-resolution stages.

Our StyleGAN-T achieves a better zero-shot MS COCO FID [140, 90] than current state-of-the-art diffusion models at a resolution of 64×64. At 256×256, StyleGAN-T halves the zero-shot FID previously achieved by a GAN but continues to trail SOTA diffusion models. The key benefits of StyleGAN-T include its fast inference speed and smooth latent space interpolation in the context of text-to-image synthesis, illustrated in Fig. 5.1 and Fig. 5.2, respectively.

5.2 StyleGAN-T

We choose StyleGAN-XL as our baseline architecture because of its strong performance in class-conditional ImageNet synthesis. In this section, we modify this baseline piece by piece, focusing on the generator (Section 5.2.1), discriminator (Section 5.2.2), and variation vs. text alignment tradeoff mechanisms (Section 5.2.3) in turn.

Throughout the redesign process, we measure the effect of our changes using zero-shot MS COCO. For practical reasons, the tests use a limited compute budget, smaller models, and a smaller dataset than the large-scale experiments in Section 5.3. We quantify sample quality using FID [90] and text alignment using CLIP score [89]. Following prior art [16], we compute the CLIP score using a ViT-g-14 model trained on LAION-2B [208].

To change the class conditioning to text conditioning in our baseline model, we embed the text prompts using a pretrained CLIP ViT-L/14 text encoder [178] and use them in place of the class embedding. Accordingly, we also remove the training-time classifier guidance. This simple conditioning mechanism matches the early text-to-image models [185, 186].

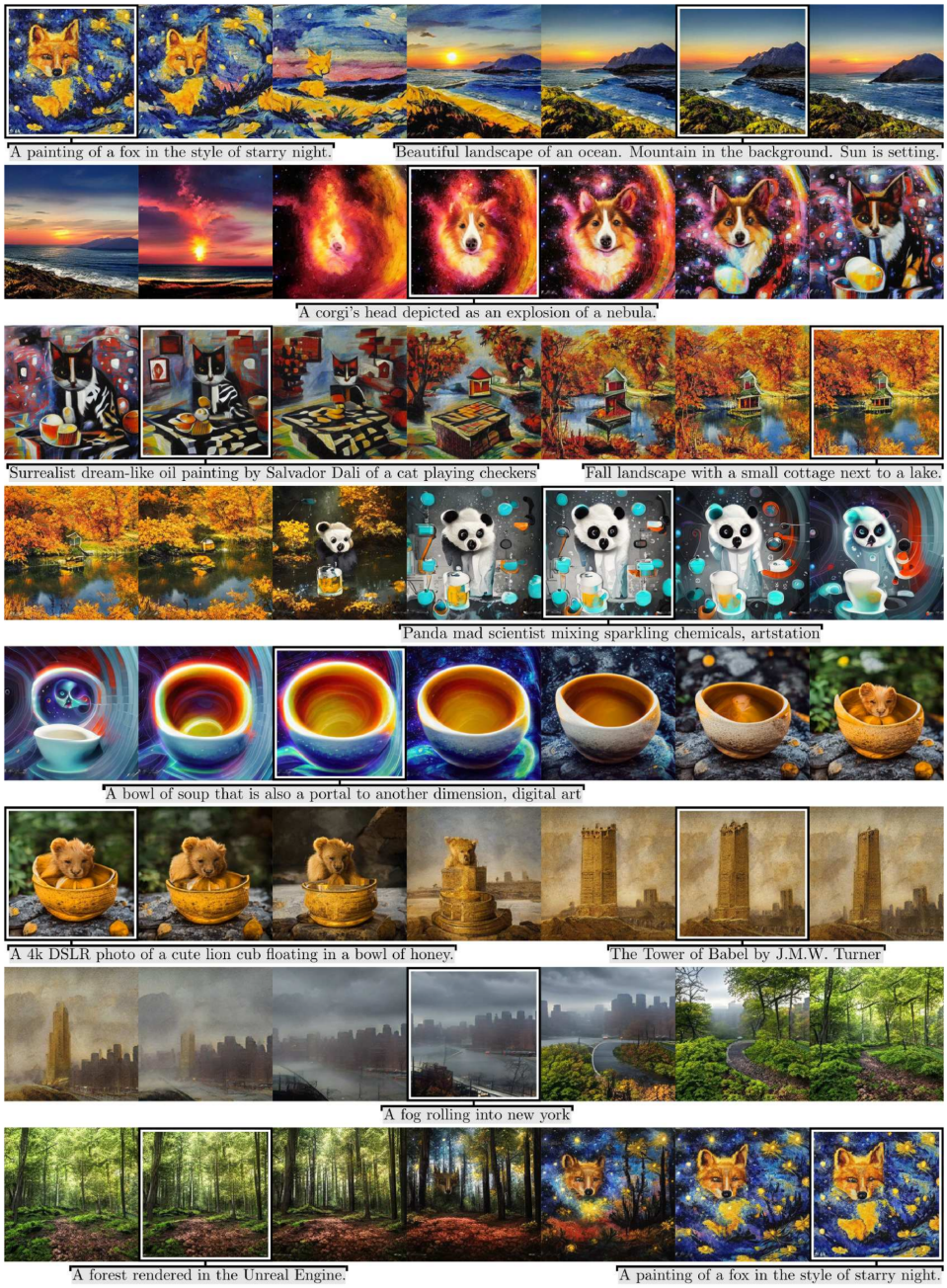

Figure 5.2: Example images and interpolations. *StyleGAN-T generates diverse samples matching the text prompt and allows for smooth interpolations between prompts, illustrated as a single continuous interpolation in scanline order. Generating these 56 samples at 512×512 takes 6 seconds on an NVIDIA RTX 3090, while a comparable grid takes up to several minutes with current diffusion models.*

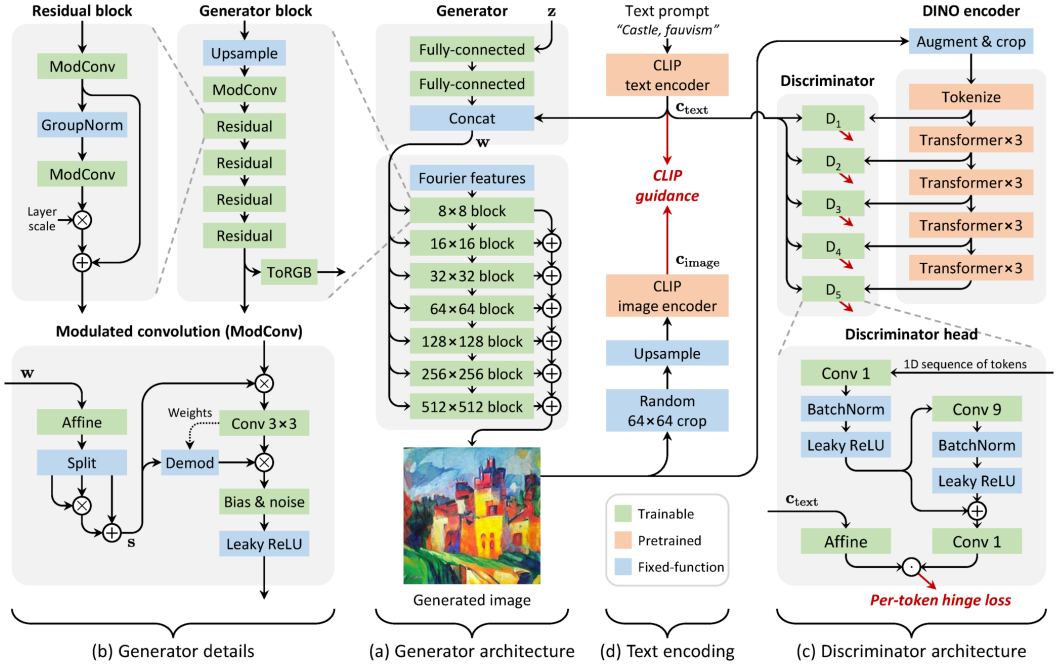

Figure 5.3: Overview of StyleGAN-T. ***(a)*** *Our generator architecture (Section 5.2.1) is closely related to StyleGAN2, with the learned constant replaced with Fourier features and conditioning applied in a slightly different place.* ***(b)*** *For each resolution, a generator block is executed and its contribution is accumulated to the image via a dedicated ToRGB layer. The generator blocks employ residual connections and a new 2^{nd} order style mechanism (Eq. 5.1).* ***(c)*** *Our discriminator (Section 5.2.2) processes the intermediate tokens of a DINO-trained vision transformer using 5 identical discriminator heads. Text conditioning is done using projection at the end.* ***(d)*** *Text prompt is embedded using CLIP and supplied to the generator and discriminator. We also employ a guidance term to further improve text alignment (Section 5.2.3).*

As shown in Table 5.1, this baseline reaches a zero-shot FID of 51.88 and CLIP score of 5.58 in our lightweight training configuration. Note that we use a different CLIP model for conditioning the generator and for computing the CLIP score, which reduces the risk of artificially inflating the results.

5.2.1 Redesigning the Generator

StyleGAN-XL uses StyleGAN3 layers to achieve translational equivariance. While equivariance can be desirable for various applications, we do not expect it to be necessary for text-to-image synthesis because none of the successful DM/ARM-based methods are equivariant. Additionally, the equivariance constraint adds computational cost and poses certain limitations to the training data that large-scale image datasets typically violate [114].

	Zero-shot FID_{30k} ↓	CLIP score ↑
StyleGAN-XL	51.88	5.58
New generator	45.10	6.02
New discriminator	26.77	9.78
$\mathcal{L}_{\text{CLIP}}$	**20.52**	**11.72**

Table 5.1: Architecture ablation. *Our architectural changes notably improve sample quality and text alignment. Here, we use the lightweight training configuration described in Section 5.3.1.*

For these reasons, we drop the equivariance requirement and switch to a StyleGAN2 backbone for the synthesis layers, including output skip connections and spatial noise inputs that facilitate stochastic variation of low-level details. The high-level architecture of our generator after these changes is shown in Fig. 5.3a. We additionally propose two changes to the details of the generator architecture (Fig. 5.3b).

Residual convolutions. As we aim to increase the model capacity significantly, the generator must be able to scale in both width and depth. However, in the basic configuration, a significant increase in the generator's depth leads to an early mode collapse in training. An important building block in modern CNN architectures [145, 53] is an easily optimizable residual block that normalizes the input and scales the output. Following these insights, we make half the convolution layers residual and wrap them by GroupNorm [246] for normalization and Layer Scale [231] for scaling their contribution. A layer scale of a low initial value of 10^{-5} allows gradually fading in the convolution layer's contribution, stabilizing the early training iterations significantly. This design allows us to increase the total number of layers considerably — by approximately $2.3\times$ in the lightweight configuration and $4.5\times$ in the final model. For fairness, we match the parameter count of the StyleGAN-XL baseline.

Stronger conditioning. The text-to-image setting is challenging because the factors of variation can vastly differ per prompt. Consider the prompts "a close-up of a face" and "a beautiful landscape." The first prompt should generate faces with varying eye color, skin color, and proportions, whereas the second should produce landscapes from different areas, seasons, and daytime. In a style-based architecture, all of this variation has to be implemented by the per-layer styles. Thus the text conditioning may need to affect the styles much more strongly than was necessary for simpler settings.

In early tests, we observed a clear tendency of the input latent $\mathbf{z}$ to dominate over the text embedding $\mathbf{c}_{\text{text}}$ in our baseline architecture, leading to poor text alignment. To remedy this, we introduce two changes that aim to amplify the role of $\mathbf{c}_{\text{text}}$. First, we let the text embeddings bypass the mapping network, following the observations by Härkönen et al. [86]. A similar design was also used in LAFITE [267], assuming that the CLIP text encoder defines an appropriate intermediate latent space for the text conditioning. We thus concatenate $\mathbf{c}_{\text{text}}$ directly to $\mathbf{w}$ and use a set of affine transforms to produce per-layer styles $\tilde{\mathbf{s}}$. Second, instead of using the resulting $\tilde{\mathbf{s}}$ to modulate the convolutions as-is, we further

split it into three vectors of equal dimension $\tilde{\mathbf{s}}_{1,2,3}$ and compute the final style vector as

$$\mathbf{s} = \tilde{\mathbf{s}}_1 \odot \tilde{\mathbf{s}}_2 + \tilde{\mathbf{s}}_3. \tag{5.1}$$

The crux of this operation is the element-wise multiplication $\odot$ that effectively turns the affine transform into a 2^{nd} order polynomial network [41, 42], increasing its expressive power. The stacked MLP-based conditioning layers in DF-GAN [226] implicitly include similar 2^{nd} order terms.

Together, our changes to the generator improve FID and CLIP score by $\sim$10%, as shown in Table 5.1.

5.2.2 Redesigning the Discriminator

We redesign the discriminator from scratch but retain Projected GAN's key ideas of relying on a frozen, pretrained feature network and using multiple discriminator heads.

Feature network. For the feature network, we choose a ViT-S [57] trained with the self-supervised DINO objective [28]. The network is lightweight, fast to evaluate, and encodes semantic information at high spatial resolution [9]. An additional benefit of using a self-supervised feature network is that it circumvents the concern of potentially compromising FID [128].

Architecture. Our discriminator architecture is shown in Fig. 5.3c. ViTs are isotropic, i.e., the representation size (tokens $\times$ channels) and receptive field (global) are the same throughout the network. This isotropy allows us to use the same architecture for all discriminator heads, which we space equally between the transformer layers. As we showed in Chapter 3, multiple heads are beneficial, and we use five heads in our design.

Our discriminator heads are minimalistic, as detailed in Fig. 5.3c, bottom. The residual convolution's kernel width controls the head's receptive field in the token sequence. We found that 1D convolutions applied on the sequence of tokens performed just as well as 2D convolutions applied on spatially reshaped tokens, indicating that the discrimination task does not benefit from whatever 2D structure remains in the tokens. We evaluate a hinge loss [137] independently for each token in every head.

For Projected GANs, we use synchronous BatchNorm [102] to provide batch statistics to the discriminator. BatchNorm is problematic when scaling to a multi-node setup, as it requires communication between nodes and GPUs. We use a variant that computes batch statistics on small virtual batches [95]. The batch statistics are not synchronized between devices but are calculated per local minibatch. Furthermore, we do not use running statistics, and thus no additional communication overhead between GPUs is introduced.

Lastly, we find that we can abstain from utilizing random projections, which we introduced in Chapter 3 as a regularization technique. Random projections were primarily aimed at obfuscating the feature space and thus balancing the adversarial game by making it more challenging for the discriminator. However, we have found through experimentation and design iterations that such obfuscation is no longer necessary with a well-conceived architecture. With only three convolutional layers per discriminator, the capacity limit in-

duces a natural regularization effect that helps prevent overfitting and maintains a balanced adversarial dynamic. This simplification retains the effectiveness of the Projected GAN paradigm and reduces complexity.

Augmentation. We apply differentiable data augmentation [265] with default parameters before the feature network in the discriminator. We use random crops when training at a resolution larger than 224×224 pixels (ViT-S training resolution).

As shown in Table 5.1, these changes significantly improve FID and CLIP score by further ∼40%. This considerable improvement indicates that a well-designed discriminator is critical when dealing with highly diverse datasets. Compared to the StyleGAN-XL discriminator, our simplified redesign is ∼2.5× faster, leading to ∼1.5× faster training.

5.2.3 Variation vs. Text Alignment Tradeoffs

Guidance [53, 93] is an essential component of current text-to-image diffusion models. It trades variation for perceived image quality in a principled way, preferring images that are strongly aligned with the text conditioning. In practice, guidance drastically improves the results; thus, we want to approximate its behavior in the context of GANs.

Guiding the generator. StyleGAN-XL uses a pretrained ImageNet classifier to provide additional gradients during training, guiding the generator toward images that are easy to classify. This method improves results significantly. In the context of text-to-image, "classification" involves captioning the images. Thus, a natural extension of this approach is to use a CLIP image encoder instead of a classifier. Following Crowson et al. [47], at each generator update, we pass the generated image through the CLIP image encoder to obtain caption $\mathbf{c}_{\text{image}}$, and minimize the squared spherical distance to the normalized text embedding $\mathbf{c}_{\text{text}}$:

$$\mathcal{L}_{\text{CLIP}} = \arccos^2(\mathbf{c}_{\text{image}} \cdot \mathbf{c}_{\text{text}}) \tag{5.2}$$

This additional loss term guides the generated distribution towards images that are captioned similarly to the input text encoding $\mathbf{c}_{\text{text}}$. Its effect is thus similar to the guidance in diffusion models. Fig. 5.3d illustrates our approach.

CLIP has been used in prior work to guide a pretrained generator during synthesis [164, 47, 144]. In contrast, we use it as a part of the loss function during training. It is important to note that overly strong CLIP guidance during training impairs FID, as it limits the distribution diversity and ultimately starts introducing image artifacts. Therefore, the weight of $\mathcal{L}_{\text{CLIP}}$ in the overall loss needs to strike a balance between image quality, text conditioning, and distribution diversity; we set it to 0.2. We further observed that guidance is helpful only up to 64×64 pixel resolution. At higher resolutions, we apply $\mathcal{L}_{\text{CLIP}}$ to random 64×64 pixel crops.

As shown in Table 5.1, CLIP guidance improves FID and CLIP scores by further ∼20%.

Guiding the text encoder. Interestingly, the earlier methods listed above that use a pretrained generator did not report encountering low-level image artifacts. We hypothesize that the frozen generator acts as a prior that suppresses them. We build on this insight to further improve the text alignment. In our primary training phase, the generator is trainable

Figure 5.4: Truncation. *Four samples for the prompt "a graphite sketch of Eva Longoria" with different random* $\mathbf{z}$*. Increasing truncation (decreasing* ψ*) improves the text alignment according to mean CLIP score per row (*$\overline{CS}$*) at the cost of lower variation.*

and the text encoder is frozen. We then introduce a secondary phase, where the generator is frozen and the text encoder becomes trainable instead. We only train the text encoder as far as the generator conditioning is concerned; the discriminator and the guidance term (Eq. 5.2) still receive $\mathbf{c}_{\text{text}}$ from the original frozen encoder. This secondary phase allows a very high CLIP guidance weight of 50 without introducing artifacts and significantly improves text alignment without compromising FID (Section 5.3.3). Compared to the primary phase, the secondary phase can be much shorter. After convergence, we continue with the primary phase.

Explicit truncation. Typically variation has been traded to higher fidelity in GANs using the truncation trick [148, 25, 115], where a sampled latent $\mathbf{w}$ is interpolated towards its mean with respect to the given conditioning input. This way, truncation pushes $\mathbf{w}$ to a higher-density region where the model performs better. In our implementation, $\mathbf{w} = [f(\mathbf{z}),\ \mathbf{c}_{\text{text}}]$, where $f(\cdot)$ denotes the mapping network, so the per-prompt mean is given by $\tilde{\mathbf{w}} = \mathbb{E}_{\mathbf{z}}[\mathbf{w}] = [\tilde{\mathbf{f}},\ \mathbf{c}_{\text{text}}]$, where $\tilde{\mathbf{f}} = \mathbb{E}_{\mathbf{z}}[f(\mathbf{z})]$. We thus implement truncation by tracking $\tilde{\mathbf{f}}$ during training and interpolating between $\tilde{\mathbf{w}}$ and $\mathbf{w}$ according to scaling parameter $\psi \in [0, 1]$ at inference time.

We illustrate the impact of truncation in Fig. 5.4. In practice, we rely on the combination of CLIP guidance and truncation. Guidance improves the model's overall text alignment, and truncation can further boost quality and alignment for a given sample, trading away some variation.

5.3 Experiments

Using the final configuration developed in Section 5.2, we scale the model size, dataset, and training time. Our final model consists of ∼1 billion parameters; we did not observe any instabilities when increasing the model size. We train on a union of several datasets amounting to 250M text-image pairs in total. We use progressive growing similar to StyleGAN-XL, except that all layers remain trainable.

5.3.1 Configuration Details

Table 5.2 lists the training and network architecture hyperparameters for our two configurations: lightweight (used for ablations) and full configuration (used for main results). Table 5.3 details the training schedules.

	Lightweight	**Full**
Generator channel base	32768	65536
Generator channel max	512	2048
Number of residual blocks per generator block	3	4
Generator parameters	75 million	1.02 billion
Text encoder parameters	123 million	123 million
Latent (z) dimension	64	64
Discriminator's feature network	DINO ViT-S/16	DINO ViT-S/16
Discriminator head's input feature space size	384	384
Discriminator head's feature space size at text conditioning	64	64
Dataset size	12M	250M
Number of GPUs	8	64
Batch size	2048	2048
Optimizer	Adam	Adam
Generator learning rate	0.002	0.002
Generator Adam betas	(0, 0.99)	(0, 0.99)
Discriminator learning rate	0.002	0.002
Discriminator Adam betas	(0, 0.99)	(0, 0.99)
EMA	0.9978	0.9978
CLIP guidance weight	0.2	0.2 (primary phase), 50 (secondary phase)
Progressive growing	No	Yes

Table 5.2: Generator, discriminator, and training hyperparameters *for the two setups used in this paper: Lightweight and Full configuration.*

Lightweight training configuration. We train using the CC12M dataset [33] at 64×64 resolution, without using progressive growing.

Full training configuration. We train using a union of several datasets: CC12m [33], CC [211], YFCC100m (filtered) [227, 218], Redcaps [51], LAION-aesthetic-6+ [208]. This amounts to a total of 250M text-image pairs. We use progressive growing similar to StyleGAN-XL, except that all layers remain trainable.

The total training time was four weeks on 64 A100 GPUs using a batch size of 2048. We first trained the primary phase for 3 weeks (resolutions up to 64×64), then the secondary phase for 2 days (text embedding), and finally the primary phase again for 5 days (resolutions up to 512×512). For comparison, our total compute budget is about a quarter of Stable Diffusion's [44].

Lightweight	**Full**
Primary Phase	Primary Phase
64x64 for 50 A100 days (25 million iterations)	16x16 for 450 A100 days (118,000 iterations)
	32x32 for 450 A100 days (78,000 iterations)
	64x64 for 450 A100 days (57,000 iterations)
	Secondary Phase
	190 A100 days (20,000 iterations)
	Primary Phase
	128x128 for 96 A100 days (10,000 iterations)
	256x256 for 70 A100 days (6,000 iterations)
	512x512 for 30 A100 days (3,000 iterations)

Table 5.3: Training schedules *for the two training configurations used in this paper. The times are listed as the number of days it would have taken on a single NVIDIA A100 GPU. An iteration corresponds to 2048 real and generated examples.*

Model	Model type	Zero-shot FID_{30k}	Speed [s]
Stable Diffusion *	Diffusion	8.40	–
eDiff-I	Diffusion	7.60	26.0
LDM *	Diffusion	7.59	–
GLIDE	Diffusion	7.40	10.9
LAFITE *	GAN	14.80	∼**0.01**
StyleGAN-T	GAN	**7.30**	0.06

* downsampled to 64×64 pixels using Lanczos – not available

Table 5.4: Comparison of FID on MS COCO 64×64. *Inference speeds are measured on an A100. For LAFITE we estimate what its speed would be at a native 64×64 resolution.*

5.3.2 Quantitative Comparison to State-of-the-Art

We use zero-shot MS COCO to compare the performance of our model to the state-of-the-art quantitatively at 64×64 pixel output resolution in Table 5.4 and 256×256 in Table 5.5. At low resolution, StyleGAN-T outperforms all other approaches in terms of output quality while being very fast to evaluate. In this test, we use the model before the final training phase, i.e., one that produces 64×64 images natively. At high resolution, StyleGAN-T still significantly outperforms LAFITE but lags behind DMs and ARMs in terms of FID.

These results lead us to two conclusions. First, GANs can match or even beat current DMs in large-scale text-to-image synthesis at low resolution. Second, a powerful super-resolution model is crucial. While FID slightly decreases in eDiff-I when moving from 64×64 to 256×256 (7.60→6.95), it currently almost doubles in StyleGAN-T. Therefore, it is evident that StyleGAN-T's superresolution stage is underperforming, causing a gap to the current state-of-the-art high-resolution results. Whether this gap can be bridged simply with additional capacity or longer training is an open question.

Model	Model type	Zero-shot FID_{30k}	Speed [s]
LDM	Diffusion	12.63	3.7
GLIDE	Diffusion	12.24	15.0
DALL·E 2	Diffusion	10.39	–
Stable Diffusion *	Diffusion	8.59	3.7
Imagen	Diffusion	7.27	9.1
eDiff-I	Diffusion	**6.95**	32.0
DALL·E	Autoregressive	27.50	–
Ernie-ViLG	Autoregressive	14.70	–
Make-A-Scene *	Autoregressive	11.84	25.0
Parti-3B	Autoregressive	8.10	6.4
Parti-20B	Autoregressive	7.23	–
LAFITE	GAN	26.94	**0.02**
StyleGAN-T *	GAN	13.90	0.10

* downsampled to 256×256 pixels using Lanczos – not available

Table 5.5: Comparison of FID on MS COCO 256×256. *Inference speeds are measured on an A100, except for Imagen and Parti that use a faster TPUv4 accelerator. The Stable Diffusion numbers are from [16, 130]; the other numbers are obtained from the respective papers or through correspondence with the authors.*

5.3.3 Evaluating Variation vs. Text Alignment

We report FID–CLIP score curves in Fig. 5.5. We compare StyleGAN-T to a strong DM baseline (CLIP-conditioned variant of eDiff-I) and a fast, distilled DM baseline (SD distilled) [149].

Using Truncation, StyleGAN-T can push the CLIP score to 0.305, successfully improving text alignment. StyleGAN-T outperforms SD-distilled in both FID and CLIP scores yet remains behind eDiff-I. Regarding speed, eDiff-I requires 32.0 seconds to generate a sample. SD-distilled is significantly faster and only needs 0.6 seconds at its best performance at eight sampling steps. StyleGAN-T beats both baselines, generating a sample in 0.1 seconds.

To isolate the impact of text encoder training, we evaluate FID–CLIP score curves in Fig. 5.6. For this experiment, we utilize the same generator network and only swap the text encoder. As the generator has been frozen in the secondary phase, it can handle both the original and fine-tuned CLIP text embeddings, as evidenced by their equal performance measured by FID. Fine-tuning the text encoder significantly improves the CLIP score without compromising FID.

5.3.4 Qualitative Results

Fig. 5.2 shows example images produced by StyleGAN-T, along with interpolations between them.

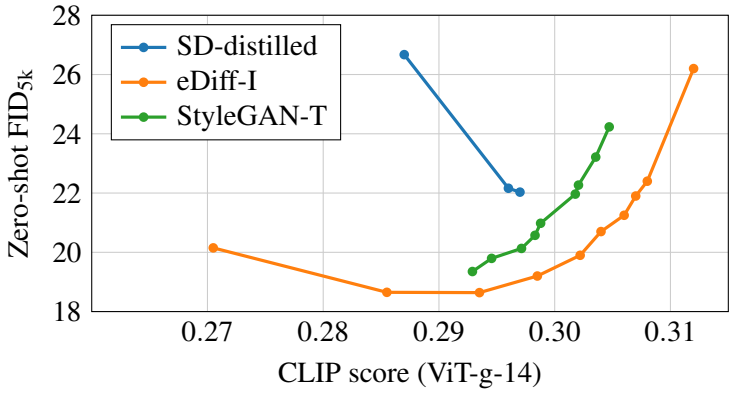

Figure 5.5: Comparing text alignment tradeoffs. *We compare FID–CLIP score curves of StyleGAN-T, distilled Stable Diffusion (SD-distilled), and eDiff-I. We report values of SD-distilled at a guidance scale of $w = 4$. For a fair comparison, we report numbers for CLIP-conditioned eDiff-I disabling additional conditioning on T5-XXL text embeddings. The models use different methods to increase the CLIP score (i.e., text alignment): StyleGAN-T decreases truncation $\psi = \{1.0 \dots 0.0\}$, SD-distilled increases the number of sampling steps $\{2, 4, 8\}$, eDiff-I increases guidance scale $w = \{0 \dots 10\}$.*

Interpolating between different text prompts is straightforward. For an image generated by an intermediate latent $\mathbf{w}_0 = [f(\mathbf{z}),\ \mathbf{c}_{\text{text0}}]$, we substitute the text condition $\mathbf{c}_{\text{text0}}$ with a new text condition $\mathbf{c}_{\text{text1}}$. We then interpolate $\mathbf{w}_0$ towards the new latent $\mathbf{w}_1 = [f(\mathbf{z}),\ \mathbf{c}_{\text{text1}}]$ as shown in Fig. 5.7. This approach is similar to DALL·E 2's text diff operation that interpolates between CLIP embeddings. Previous work for manipulating GAN-generated images [171] typically discovers these latent directions via a training process that needs to be repeated per prompt and is, therefore, expensive. Meaningful latent directions are a built-in property of our model, and no extra training is needed.

By appending different styles to a prompt, StyleGAN-T can generate a wide variety of styles as shown in Fig. 5.8. Subjects tend to be aligned for a fixed latent $\mathbf{z}$.

Fig. 5.9 shows additional examples of truncation. Fig. 5.10 shows qualitative comparisons to Latent Diffusion [191], Stable Diffusion [191], DALL·E 2 [183]. We use the same prompts as in the truncation study.

5.4 Discussion

Similarly to DALL·E 2, which also uses CLIP as the underlying language model, StyleGAN-T sometimes struggles in terms of binding attributes to objects as well as producing coherent text in images (Fig. 5.11). Using a larger language model would likely resolve this issue at the cost of slower runtime [194, 16].

Guidance via CLIP loss is vital for good text alignment, but high guidance strength results in image artifacts. A possible solution could be to retrain CLIP on higher-resolution data that does not suffer from aliasing or other image quality issues. In this context, the

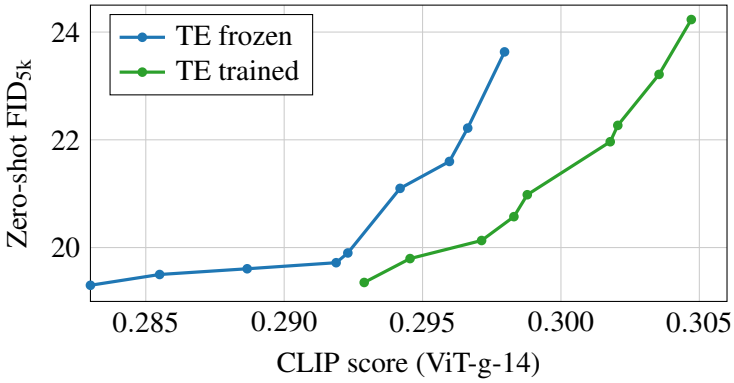

Figure 5.6: Text encoder training. *Training the CLIP text encoder (TE) pushes the entire FID–CLIP score curve to the right, hence, increasing overall text alignment.*

conditioning mechanism in the discriminator may also be worth revisiting.

Truncation improves text alignment but differs from guidance in diffusion models in two important ways. While truncation is always towards a single mode, guidance can at least theoretically be arbitrarily multi-modal. Also, truncation sharpens the distribution before the synthesis network, which can reshape the distribution in arbitrary ways, thus, possibly undoing any prior sharpening. Therefore, alternative methods to truncation might further improve the results.

Improved super-resolution stages (i.e., high-resolution layers) through higher capacity and longer training are obvious avenues for future work.

Methods for "personalizing" diffusion models have become popular [193, 73]. They finetune a pretrained model to associate a unique identifier with a given subject, allowing it to synthesize novel images of the same subject in novel contexts. Such approaches could be similarly applied to GANs.

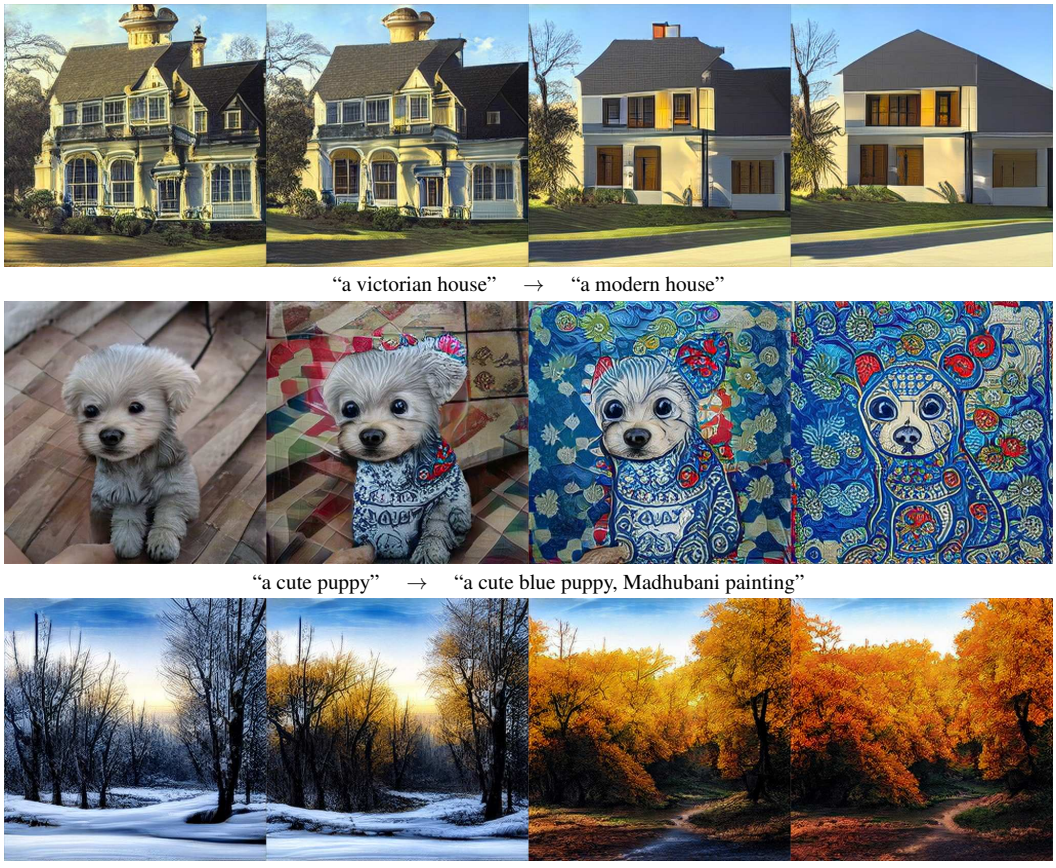

Figure 5.7: Latent manipulation. *Samples (first column) can be manipulated by following semantic directions in latent space.*

Figure 5.8: Styles. *Samples generated by StyleGAN-T for a fixed random seed and the caption "astronaut, {X}", where X is denoted below each image.*

"A painting of a fox in the style of starry night"

"A surrealist dream-like oil painting by Salvador Dalí of a cat playing checkers"

"Robots meditating in a vipassana retreat"

"A teddy bear on a skateboard in times square"

"A still of Kermit The Frog in WALL-E (2008)"

"A transformer robot with legs and arms made out of vegetation and leaves"

Figure 5.9: Additional truncation grids. *We show samples for 6 different prompts and 5 different random latents, shared between the prompts. Increasing truncation (decreasing ψ), improves the text alignment according to mean CLIP score per row, $\overline{CS}$, at the cost of lower variation.*

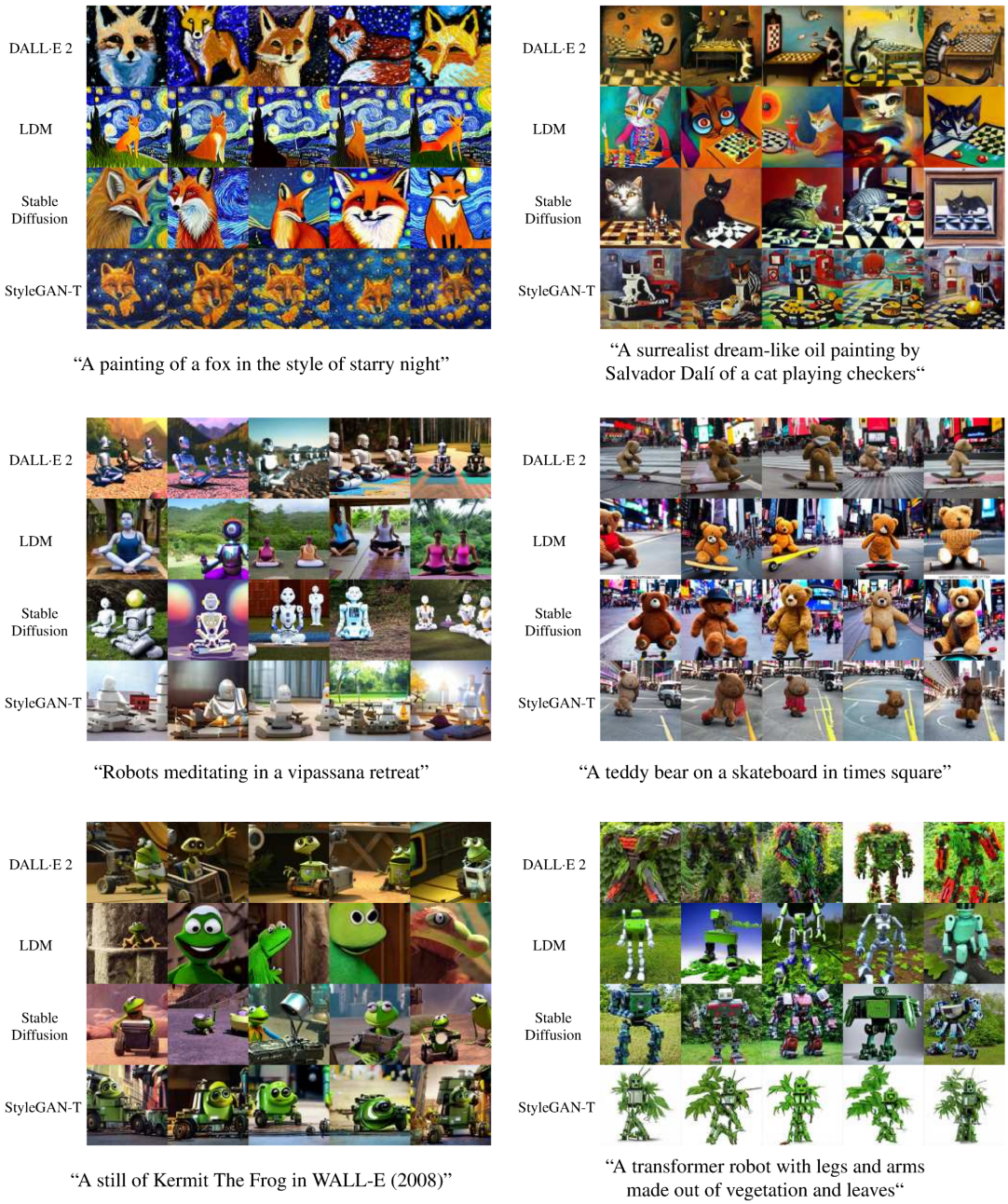

Figure 5.10: Qualitative Comparisons. *We show samples for 6 different prompts and 5 different random latents, shared between the prompts. For StyleGAN-T, we set* $\psi = 0.6$. *LDM and Stable Diffusion utilize 250 and 50 sampling steps, respectively, utilizing the DDIM / PLMS sampler [143]. For DALL·E 2, we generate images via the official DALL·E service [168].*

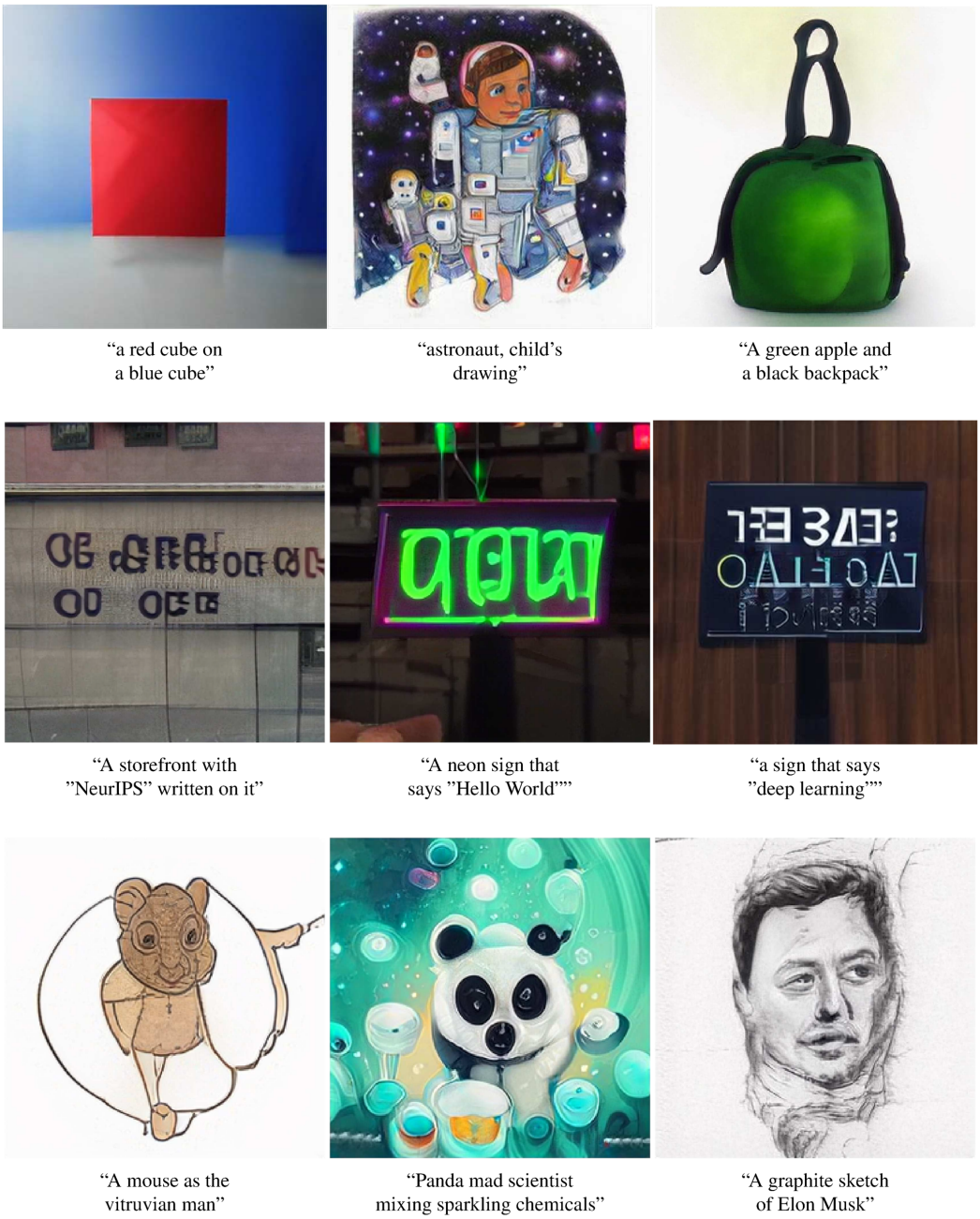

Figure 5.11: Failure cases. *StyleGAN-T can fail to bind attributes to objects or generate separate entities (top row) and to produce coherent text (middle row). Furthermore, the model struggles at high-resolution, resulting in samples with low details and text coherence (bottom row).*

6 Quantifying Progress: Projected GAN to StyleGAN-XL to StyleGAN-T

In this chapter, we focus on quantifying the advancements made in our research, from the introduction of Projected GANs (Chapter 3) to the development of StyleGAN-XL (Chapter 4) and StyleGAN-T (Chapter 5). Throughout the progression of our research, we consistently strived to improve both the architecture of the models and the training methods applied. Concurrently, the training settings have progressively grown in scale and complexity. In the subsequent sections, we delve into a comparative study of these approaches, analyzing each model's architecture and training configuration. Notably, we ensure a fair comparison by applying all methods to the same dataset.

6.1 Comparing Configurations

We compare generator, discriminator, guidance, and training hyperparameters in Table 6.1.

Generator. The type of layers in the generator evolve from standard convolutions in FastGAN to alias-free modulated convolutions in StyleGAN3 and, eventually, back to standard modulated convolutions in StyleGAN2. This transition was guided by the unique benefits each layer type provides. We initially opted for StyleGAN2 to obtain alias-free generation with the added advantage of modulated convolutions, which improve inversion and controllability. For StyleGAN-T, we incorporate StyleGAN2 layers as they are computationally cheaper than StyleGAN3 layers, albeit without the added benefit of alias-free generation. A notable observation is the benefit of reducing the latent dimension. This adjustment enables the training of style-based generators within the Projected GAN paradigm.

As our research progresses, there is a consistent increase in the size and depth of our generator models. This trend peaks with StyleGAN-T, which incorporates up to 1 billion parameters, directly paralleling the growth of our datasets. Finally, the conditioning mechanism is adapted to the respective training settings. We transition from unconditional generation to class-conditional to text-conditional generation.

Discriminator & Guidance. We consistently leverage hinge loss throughout our work, following [142], as our experiments did not identify any other loss functions as superior. Even though differentiable augmentation for GANs [265] was developed to boost performance on small datasets, we observe in Chapter 4 that it is also beneficial in large-scale settings. We posit that data augmentation helps prevent the discriminator from focusing on imperceptible, high-frequency signals, similar to adversarial attacks, as noted [144].

In designing the discriminator architecture, we find that random projections are not needed with the right architecture. As conjectured in Chapter 5, this is primarily due to

the low-capacity architecture implicitly constraining the discriminator sufficiently. The discriminator's feature network started from a basic CNN. In StyleGAN-XL, we add a ViT for richer feedback. For StyleGAN-T, we switch to a ViT trained with a self-supervised objective. As the isotropic architecture of the ViT facilitates a controlled search for the optimal discriminator architecture, we are able to run more targeted experiments. With the new architecture, we observe that a single ViT suffices as the feature network. Since the performance among different feature networks is similar, we opt for a self-supervised feature network, avoiding the risk of potentially compromising FID [128]. As observed in the discriminator parameter counts, there is no correlation between larger feature networks and improved generative performance, as noted in Chapter 3. We supplement the discriminator's signal with an additional discriminative model in conditional settings. For ImageNet, we use a classifier; in the text-conditional setting, we leverage CLIP.

Training. The datasets in our experiments evolve from small or large unimodal datasets to large multimodal datasets such as ImageNet, and eventually to even larger text-conditional datasets. This increase in complexity and diversity is mirrored by a corresponding increase in the number of GPUs required for training the models. Despite these changes in the datasets and hardware requirements, the settings for the optimizer remain consistent throughout our experiments. One significant observation is the importance of large batch sizes in multimodal settings. Similar to [25], we find that a larger batch size allows each batch to better reflect the dataset's diversity.

Finally, we observe that reintroducing progressive growing in StyleGAN-XL is crucial for handling multimodal settings. This technique, first described in [111], enables the model to learn to generate images of increasing complexity gradually.

6.2 Comparing Performance

We evaluated Projected GAN in an unconditional setting. In contrast, StyleGAN-XL and StyleGAN-T are applied to class- and text-conditional data. To facilitate a fair comparison, we modify these models to also work unconditionally. For StyleGAN-XL, we achieve this by removing the conditioning and classifier guidance, as described in Chapter 4. This modification allows StyleGAN-XL to generate data without needing specific class conditions. Similarly, for StyleGAN-T, we remove the text-encoder and CLIP guidance. This adjustment allows StyleGAN-T to generate images without reliance on a specific textual description. These modifications are straightforward and enable direct comparison between all three models.

Setup. We select FFHQ [116] as the benchmark dataset. For fairness, we endow all generators with roughly the same number of parameters ($\sim$ 30M). For all models, we train until FID converges.

Results. We show uncurated sample grids of all models Fig. 6.1. In the progression from Projected GAN to StyleGAN-XL to StyleGAN-T, the sample quality and diversity clearly increase visually and as measured by FID.

	Projected GAN	**StyleGAN-XL**	**StyleGAN-T**
G layer type	FastGAN	StyleGAN3	StyleGAN2
z dimension	256	64	64
w dimension	-	512	512
G depth (# of convolution layers)	11	37	63
G channel base	65536	65536	65536
G channel max	2048	1024	2048
G model size	80M	168M	1020M
Conditioning	-	Class	Text
Adversarial loss type	Hinge	Hinge	Hinge
Differentiable Augmentation	True	True	True
Random Projections	True	True	False
D's feature network	Effnet-lite0	Effnet-lite0 & DeiT ViT-B/16	DINO ViT-S/16
D model size (feature network)	3M	90M	21M
D model size (random projections)	0.3M	7M	-
D model size (trainable)	14M	35M	13M
Guidance	-	CLF guidance	CLIP guidance
Dataset size	300 to 3M	1.3M	250M
Number of GPUs	8	8 to 32	64
Optimizer	Adam	Adam	Adam
Batch size	64	2048	2048
G learning rate	0.0025	0.0025	0.0025
D learning rate	0.0025	0.0025	0.0025
β_1	0.0	0.0	0.0
β_2	0.99	0.99	0.99
# D updates per G update	1	1	1
Progressive growing	False	True	True

Table 6.1: Generator, discriminator, guidance, and training hyperparameters *for the introduced approaches. We compare generators for output resolution 512x512 pixels.*

6.3 Discussion

In the unconditional setting, the discriminator is the main distinguishing factor. Therefore, our experiments on FFHQ underscore the critical role of the discriminator within the GAN framework. The discriminator is responsible for achieving high sample quality and comprehensive distribution coverage. In the context of conditional generation, the introduction of guidance can augment the feedback from the discriminator, thereby improving alignment with the condition.

The Projected GAN paradigm can leverage ongoing advancements in representation learning. Our progression of feature networks – from EfficientNet, to EfficientNet coupled with ViT, to DINO ViT – demonstrates this adaptability. Nevertheless, the quest for identifying the optimal features for the discriminator remains ongoing.

Our findings also underline the importance of progressive growing in efficiently training deep networks. Although Multi-Scale Gradients (MSG) pose a plausible alternative approach [110], progressive growing presents additional computational advantages, such as higher throughput at lower resolution [111].

Lastly, as evident from the increasing parameter count of the generator, our contributions

Figure 6.1: Uncurated Results for FFHQ *(256²). For a fair comparison, we show untruncated samples. The images are selected randomly given one global random seed. We recommend zooming in for comparison.*

have made it feasible to train much larger models than previously possible, pushing the boundary of what GANs can achieve.

7 Conclusion

In this thesis, we have systematically studied GANs to better understand their training efficiency, scalability, and performance in various contexts. Our research on Projected GANs highlights the importance of pretrained feature networks in improving stability and data efficiency during GAN training. Building on the success of Projected GANs, we aimed to incorporate their advantages into StyleGAN and further investigate the potential of large-scale training of GANs. We subsequently focused on adapting StyleGAN to achieve competitive performance on ImageNet. After successfully scaling GANs on ImageNet, we explored their application in a larger-scale setting, specifically text-to-image synthesis.

This work presents our pioneering attempt at constructing a GAN foundation model. While our model exhibits comparable performance to the current state-of-the-art at lower resolutions, a performance gap persists for high-resolution synthesis. However, our work identifies a promising pathway towards achieving state-of-the-art performance through increased computational resources, the enhancement of our superresolution model, and further refinement of Projected GAN paradigm.

In the following, we identify current limitations, suggest potential improvements, and consider long-term research directions for GANs, particularly in large-scale applications.

7.1 Limitations and Future Work

There are still several limitations to GANs, some of which can be addressed through ongoing research, while others are intrinsic to the model family. For more specific discussions on limitations and future directions with respect to individual contributions, we refer readers to the conclusion sections of the respective chapters.

Stable Training. Training GANs can be particularly challenging due to factors such as saddle point optimization [79], non-overlapping manifolds [12], and mode collapse [13]. Moreover, loss metrics are generally less informative, resulting in the reliance on proxy metrics to monitor training progress [90]. However, stability can be significantly enhanced by employing regularization techniques like R1 regularization [151] and adopting the Projected GAN paradigm. In our large-scale experiments with StyleGAN-T, we encountered no stability issues, illustrating the efficacy of these strategies. Consequently, stability, often considered the most pressing problem when training GANs, may no longer be a major concern for the image synthesis setting.

Iterative Refinement. In their standard configuration, GANs generate data in a single pass, providing no opportunity to correct errors. In contrast, diffusion models and autoregressive models benefit from reusing the model, enabling improvements when inference time and cost are not limiting factors. While one possibility involves adapting GANs to

run multiple times [10], diffusion models offer a more principled approach to achieving this goal. As a result, diffusion models are likely to maintain their superiority when it comes to models that perform several iterations.

Diversity-Quality Tradeoff. In the case of StyleGAN-T, we discovered that truncation can effectively balance image variation and text alignment. However, truncation is less effective than classifier-free guidance [93] regarding both sample quality and text alignment. A more advanced method is needed, which either enhances truncation or emulates classifier-free guidance.

Domains Beyond Image Synthesis. The focus of this thesis is on image synthesis tasks. Adapting the proposed approaches, such as Projected GANs, to new domains might prove challenging. In fields like adversarial 3D-aware synthesis [209, 210, 31] and speech generation [122], a non-pretrained discriminator remains the standard as of yet. Furthermore, GANs are less suitable for text generation [8]. Additional research is required to determine the practical application of GANs across various domains, as their utility may vary depending on the area of focus.

7.2 The Future of GANs in the Generative Model Landscape

In the evolving landscape of generative models, three prominent model families currently stand out: GANs, Diffusion Models (DMs), and Autoregressive Models (ARMs). For natural language processing (NLP), ARMs have emerged as the dominant approach, as evidenced by models like GPT-3 [27] and LLaMa [232]. While DMs have made their initial forays into NLP [135, 77], GANs are considered less useful for text generation [8]. However, when it comes to generating visual data, the situation differs. It has been shown that both ARMs and DMs can be trained at scale for image synthesis, and our work demonstrates that this is also possible with GANs.

GANs will likely remain relevant in the generative model landscape due to their single-shot generation capability, architectural flexibility, and versatile latent space. Although GANs can be challenging to train, the potential payoff is substantial. GANs will likely find continued success and adoption in various domains as they overcome their training challenges and further demonstrate their unique strengths. The following discussion will focus on the role of GANs for generating visual data.

Large-Scale Image Synthesis. The role of GANs in large-scale image synthesis is evolving, with recent advancements like our proposed StyleGAN-T showing that GANs can close the performance gap with other generative models. Scaling laws for GANs may be further explored in future research by increasing the model size and allocating more compute resources. Moreover, as GANs can generate high-quality images significantly faster than diffusion models, deploying such models could save substantial computational costs. Currently, the research community is primarily focused on diffusion models due to their easier training process. Furthermore, considerable research is devoted to utilizing and investigating high-quality pretrained models. In the context of GenAI, this trend first emerged with StyleGAN and its various iterations [115, 116, 113, 114], inspiring numerous follow-

up works that either explored pretrained checkpoints [187, 87, 171] or adapted them for new applications [17]. The open-source release of generative foundation models, such as Stable Diffusion [191], has triggered a Cambrian explosion of applications [258, 245, 75, 202, 83, 147]. A GAN of similar quality could have a comparable impact, reinvigorating GAN research and driving new developments in large-scale image synthesis.

Concrete, possible applications of GANs for general image synthesis include:

- As a standalone foundation model for general large-scale text-based image synthesis, i.e., generating new images from scratch.
- Image-to-image translation through adapters, i.e., an additional encoder trained for a given foundation model. A similar approach works well for DMs [258].
- As a backbone for personalized image synthesis, similar to DreamBooth for DMs [193]. DreamBooth for GANs will unlock much faster generation of personalized content.
- As initial seed image generator for a DM. The GAN quickly generates many samples of lower quality, which can be selected by a human practitioner and fed into a more powerful but slower DM.

Generative AI Systems. The role of GANs, and more broadly, adversarial losses, in generative AI systems is multifaceted. An adversarial loss can be combined with different objectives, such as training the autoencoder of VQ-GANs [67] or latent diffusion models [191]. On the other hand, the diffusion steps of a forward diffusion chain can be employed to enhance GAN training [243]. Another approach to harnessing the power of GANs in generative systems might involve generating synthetic data with DMs and subsequently distilling this data into a GAN. In this setup, DMs offer raw generative capability, while GANs facilitate rapid inference. Another potential application entails using GANs to refine the samples generated by diffusion models. In this scenario, GANs serve as post-processing tools to improve the image quality of DMs and eliminate artifacts, yielding more realistic results.

In summary, potential applications for GANs in GenAI systems are

- Employed as an adversarial loss component during auto-encoder training, GANs enhance detail fidelity in high-resolution outputs.
- For distilling high-quality DM samples in narrow domains on which GANs work well. A prominent example is FFHQ, i.e., a human portrait dataset on which GANs excel. The DM could generate a synthetic dataset on which the GAN is trained and enables faster inference.
- As a post-processing tool for other generative models. GANs can do artifact removal and superresolution potentially as well as other models, but much faster.

Special use-cases. GANs exhibit several unique advantages, making them suitable for specific use cases where other generative models might struggle. Real-time performance, for

example, can be challenging to achieve with DMs as there are limits to how much they can be sped up. Moreover, antialiasing can be difficult for DMs, as suppressing aliasing requires intricately designed architectures and assumes a hierarchical synthesis process inherent in GANs [114]. GANs are also better suited for video editing applications. They offer much faster processing times and make it easier to achieve temporal consistency, even without training on video data [250]. This advantage enables seamless editing. Lastly, GANs have proven valuable in specialized areas, such as scientific or medical data generation, where foundation models might be less useful. For instance, GANs have been successfully applied for generating solar images [39] and medical scans [101], achieving performance on par with other generative models while offering faster training and sampling times.

To summarize, GANs shine in special use-cases:

- For applications requiring real-time performance, such as interactive sample generation and control, real-time image-to-image translation, e.g., as a realism filter for video games.
- GANs are especially well-suited for video editing, offering faster processing times and better temporal consistency, even without training on video data as demonstrated by [250].
- In specialized domains such as scientific or medical data generation, GANs perform comparably to foundation models but with faster training and sampling times.

In summary, the unique strengths of GANs ensure their continued relevance in the generative model landscape, especially in specialized use cases where their capabilities shine.

A Credits

In the next section, we detail the individual contributions for the three research projects in this thesis. These results emerged from collective efforts, and precise role attribution may be approximate. All authors significantly contributed to each publication, enabling the success of the projects.

Projected GANs Converge Faster (Chapter 3) Axel Sauer (AS) conceived the initial idea of utilizing pretrained representations for GAN training and led the entire project. Kashyap Chitta (KC) contributed to baseline implementation, figure creation, manuscript drafting, proofreading, and project direction discussions. Jens Müller (JM) also contributed to the baseline implementation, figure creation, manuscript drafting, proofreading, and project direction. Andreas Geiger (AG) participated in manuscript writing, proofreading, figure creation, and project direction discussions.

StyleGAN-XL: Scaling StyleGAN to Large Diverse Datasets (Chapter 4) The concept of scaling StyleGAN to diverse datasets originated from discussions between AS and Katja Schwarz (KS). AS led the project, overseeing all aspects. KS contributed to experimental evaluation, figure creation, manuscript drafting, proofreading, and project direction discussions. AG participated in manuscript writing, proofreading, figure creation, and project direction discussions.

StyleGAN-T: Unlocking the Power of GANs for Fast Large-Scale Text-to-Image Synthesis (Chapter 5) The concept of training StyleGAN for large-scale text-to-image synthesis was jointly conceived by AS, Tero Karras (TK), and Timo Aila (TA). AS led the entire project. TK, TA, AG, and Samuli Laine (SL) collaborated on figure creation, manuscript drafting, proofreading, and project direction discussions.

Bibliography

[1] R. Abdal, Y. Qin, and P. Wonka. "Image2StyleGAN: How to Embed Images Into the StyleGAN Latent Space?" In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2019.

[2] R. Abdal, P. Zhu, N. J. Mitra, and P. Wonka. "StyleFlow: Attribute-Conditioned Exploration of StyleGAN-Generated Images Using Conditional Continuous Normalizing Flows". In: *ACM Trans. on Graphics* 40.3 (2021).

[3] R. Abdal, P. Zhu, N. J. Mitra, and P. Wonka. "StyleFlow: Attribute-conditioned Exploration of StyleGAN-Generated Images using Conditional Continuous Normalizing Flows". In: *ACM Trans. on Graphics* (2021).

[4] Adobe. *Adobe Firefly*. https://www.adobe.com/sensei/generative-ai/firefly.html. Accessed: April 18, 2023.

[5] Y. Alaluf, O. Patashnik, and D. Cohen-Or. "ReStyle: A Residual-Based StyleGAN Encoder via Iterative Refinement". In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)* (2021).

[6] Y. Alaluf, O. Patashnik, Z. Wu, A. Zamir, E. Shechtman, D. Lischinski, and D. Cohen-Or. "Third Time's the Charm? Image and Video Editing with StyleGAN3". In: *arXiv.org* (2022).

[7] I. Albuquerque, J. Monteiro, T. Doan, B. Considine, T. Falk, and I. Mitliagkas. "Multi-objective training of generative adversarial networks with multiple discriminators". In: *Proc. of the International Conf. on Machine learning (ICML)*. 2019.

[8] D. Alvarez-Melis, V. Garg, and A. T. Kalai. "Why GANs are overkill for NLP". In: *arXiv.org* (2022).

[9] S. Amir, Y. Gandelsman, S. Bagon, and T. Dekel. "Deep VIT features as dense visual descriptors". In: *CoRR* abs/2112.05814 (2021).

[10] D. Arad Hudson and L. Zitnick. "Compositional transformers for scene generation". In: *Advances in Neural Information Processing Systems* 34 (2021).

[11] M. Arjovsky and L. Bottou. "Towards principled methods for training generative adversarial networks". In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2017.

[12] M. Arjovsky and L. Bottou. "Towards principled methods for training generative adversarial networks". In: *Proc. of the International Conf. on Learning Representations (ICLR)* (2017).

[13] M. Arjovsky, S. Chintala, and L. Bottou. "Wasserstein generative adversarial networks". In: *International conference on machine learning*. PMLR. 2017.

[14] S. Azizi, S. Kornblith, C. Saharia, M. Norouzi, and D. J. Fleet. “Synthetic Data from Diffusion Models Improves ImageNet Classification”. In: *arXiv.org* (2023).

[15] J. L. Ba, J. R. Kiros, and G. E. Hinton. “Layer normalization”. In: *arXiv.org* (2016).

[16] Y. Balaji, S. Nah, X. Huang, A. Vahdat, J. Song, K. Kreis, M. Aittala, T. Aila, S. Laine, B. Catanzaro, et al. “eDiff-I: Text-to-Image Diffusion Models with an Ensemble of Expert Denoisers”. In: *CoRR* abs/2211.01324 (2022).

[17] A. H. Bermano, R. Gal, Y. Alaluf, R. Mokady, Y. Nitzan, O. Tov, O. Patashnik, and D. Cohen-Or. “State-of-the-Art in the Architecture, Methods and Applications of StyleGAN”. In: *Computer Graphics Forum*. Vol. 41. 2. Wiley Online Library. 2022.

[18] A. H. Bermano, R. Gal, Y. Alaluf, R. Mokady, Y. Nitzan, O. Tov, O. Patashnik, and D. Cohen-Or. “State-of-the-Art in the Architecture, Methods and Applications of StyleGAN”. In: *CoRR* abs/2202.14020 (2022).

[19] M. Binkowski, D. J. Sutherland, M. Arbel, and A. Gretton. “Demystifying MMD GANs”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2018.

[20] S. Blake. “Flawless AI Dubbing: The Future of Seamless Voiceovers in Film”. In: *Dot.LA* (May 2021). Accessed: 2023-04-17. URL: https://dot.la/flawless-ai-dubbing-2652865135.html.

[21] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, et al. “On the opportunities and risks of foundation models”. In: *arXiv.org* (2021).

[22] A. Borji. “Pros and cons of gan evaluation measures”. In: *Computer Vision and Image Understanding* (2019).

[23] N. Brandes, D. Ofer, Y. Peleg, N. Rappoport, and M. Linial. “ProteinBERT: a universal deep-learning model of protein sequence and function”. In: *Bioinformatics* 38.8 (2022).

[24] A. Brock, J. Donahue, and K. Simonyan. “Large Scale GAN Training for High Fidelity Natural Image Synthesis”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2019.

[25] A. Brock, J. Donahue, and K. Simonyan. “Large scale GAN training for high fidelity natural image synthesis”. In: *Proc. of the International Conf. on Learning Representations (ICLR)* (2019).

[26] T. Brooks, A. Holynski, and A. A. Efros. “Instructpix2pix: Learning to follow image editing instructions”. In: *arXiv.org* (2022).

[27] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. “Language models are few-shot learners”. In: *Advances in neural information processing systems* 33 (2020).

[28] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. "Emerging properties in self-supervised vision transformers". In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2021.

[29] A. Casanova, M. Careil, J. Verbeek, M. Drozdzal, and A. Romero-Soriano. "Instance-Conditioned GAN". In: *Advances in Neural Information Processing Systems (NEURIPS)*. 2021.

[30] L. Chai, J. Zhu, E. Shechtman, P. Isola, and R. Zhang. "Ensembling With Deep Generative Views". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. Computer Vision Foundation / IEEE, 2021.

[31] E. R. Chan, C. Z. Lin, M. A. Chan, K. Nagano, B. Pan, S. De Mello, O. Gallo, L. J. Guibas, J. Tremblay, S. Khamis, et al. "Efficient geometry-aware 3D generative adversarial networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022.

[32] T. Chang and C. Lu. "TinyGAN: Distilling BigGAN for Conditional Image Generation". In: *Proc. of the Asian Conf. on Computer Vision (ACCV)*. Ed. by H. Ishikawa, C. Liu, T. Pajdla, and J. Shi. Lecture Notes in Computer Science. 2020.

[33] S. Changpinyo, P. Sharma, N. Ding, and R. Soricut. "Conceptual 12M: Pushing Web-Scale Image-Text Pre-Training To Recognize Long-Tail Visual Concepts". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[34] S. Changpinyo, P. Sharma, N. Ding, and R. Soricut. "Conceptual 12m: Pushing web-scale image-text pre-training to recognize long-tail visual concepts". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021.

[35] M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, and I. Sutskever. "Generative pretraining from pixels". In: *International conference on machine learning*. 2020.

[36] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. "Evaluating large language models trained on code". In: *arXiv.org* (2021).

[37] X. Chen, N. Mishra, M. Rohaninejad, and P. Abbeel. "Pixelsnail: An improved autoregressive generative model". In: *Proc. of the International Conf. on Machine learning (ICML)*. 2018.

[38] X. Chen, H. Fan, and R. G. K. He. *Improved baselines with momentum contrastive learning*. 2020. arXiv.org: 2003.04297.

[39] M. Cherti, A. Czernik, S. Kesselheim, F. Effenberger, and J. Jitsev. "A Comparative Study on Generative Models for High Resolution Solar Observation Imaging". In: *arXiv.org* (2023).

[40] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha. "Stargan v2: Diverse image synthesis for multiple domains". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[41] G. Chrysos, S. Moschoglou, G. Bouritsas, J. Deng, Y. Panagakis, and S. Zafeiriou. "Deep Polynomial Neural Networks". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[42] G. G. Chrysos and Y. Panagakis. "CoPE: Conditional image generation using Polynomial Expansions". In: *Advances in Neural Information Processing Systems (NEURIPS)*. 2021.

[43] E. Collins, R. Bala, B. Price, and S. Süsstrunk. "Editing in Style: Uncovering the Local Semantics of GANs". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. Computer Vision Foundation / IEEE, 2020.

[44] CompVis. *Stable Diffusion Model Card*. 2022. URL: https://github.com/CompVis/stable-diffusion/blob/main/Stable_Diffusion_v1_Model_Card.md.

[45] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[46] A. Creswell and A. A. Bharath. "Inverting the Generator of a Generative Adversarial Network". In: *IEEE Trans. Neural Networks Learn. Syst.* (2019).

[47] K. Crowson, S. Biderman, D. Kornis, D. Stander, E. Hallahan, L. Castricato, and E. Raff. "VQGAN-CLIP: Open domain image generation and editing with natural language guidance". In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2022.

[48] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "Imagenet: A large-scale hierarchical image database". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2009.

[49] J. Deng, W. Dong, R. Socher, L.-j. Li, K. Li, and L. Fei-fei. "Imagenet: A large-scale hierarchical image database". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2009.

[50] K. Desai and J. Johnson. "Virtex: Learning visual representations from textual annotations". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[51] K. Desai, G. Kaul, Z. Aysola, and J. Johnson. "RedCaps: Web-curated image-text data created by the people, for the people". In: *Proc. Neurips (Datasets and Benchmarks)*. 2021.

[52] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv.org* (2018).

[53] P. Dhariwal and A. Nichol. "Diffusion Models Beat GANs on Image Synthesis". In: (2021).

[54] P. Dhariwal and A. Nichol. "Diffusion Models beat Gans on Image Synthesis". In: *Advances in Neural Information Processing Systems (NEURIPS)* (2021).

[55] F. Di Mattia, P. Galeone, M. De Simoni, and E. Ghelfi. “A survey on gans for anomaly detection”. In: *arXiv.org* (2019).

[56] T. Doan, J. Monteiro, I. Albuquerque, B. Mazoure, A. Durand, J. Pineau, and R. D. Hjelm. “On-line adaptative curriculum learning for gans”. In: *Proc. of the Conf. on Artificial Intelligence (AAAI)*. 2019.

[57] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2021.

[58] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2021.

[59] A. Dosovitskiy and T. Brox. “Generating images with perceptual similarity metrics based on deep networks”. In: *Advances in Neural Information Processing Systems (NIPS)*. 2016.

[60] A. Dosovitskiy and T. Brox. “Inverting Visual Representations with Convolutional Networks”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[61] D. Dowson and B. Landau. “The Fréchet distance between multivariate normal distributions”. In: *Journal of multivariate analysis* (1982).

[62] N. Drobyshev, J. Chelishev, T. Khakhulin, A. Ivakhnenko, V. Lempitsky, and E. Zakharov. “Megaportraits: One-shot megapixel neural head avatars”. In: *arXiv.org* (2022).

[63] I. Durugkar, I. Gemp, and S. Mahadevan. “Generative multi-adversarial networks”. In: *arXiv.org* (2016).

[64] I. Durugkar, I. Gemp, and S. Mahadevan. “Generative multi-adversarial networks”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2017.

[65] P. Esser, J. Chiu, P. Atighehchian, J. Granskog, and A. Germanidis. “Structure and content-guided video synthesis with diffusion models”. In: *arXiv.org* (2023).

[66] P. Esser, R. Rombach, and B. Ommer. “Taming Transformers for High-Resolution Image Synthesis”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[67] P. Esser, R. Rombach, and B. Ommer. “Taming transformers for high-resolution image synthesis”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021.

[68] H. Face. *Hugging Face Spaces*. Accessed: 2023-04-17. 2021. URL: https://huggingface.co/spaces.

[69] J. D. Foley, F. D. Van, A. Van Dam, S. K. Feiner, and J. F. Hughes. *Computer graphics: principles and practice*. Vol. 12110. Addison-Wesley Professional, 1996.

[70] A. Fregin, J. Müller, U. Krebel, and K. Dietmayer. "The DriveU Traffic Light Dataset: Introduction and Comparison with Existing Datasets". In: *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*. IEEE, 2018.

[71] S. Y. Gadre, G. Ilharco, A. Fang, J. Hayase, G. Smyrnis, T. Nguyen, R. Marten, M. Wortsman, D. Ghosh, J. Zhang, et al. "DataComp: In search of the next generation of multimodal datasets". In: *arXiv.org* (2023).

[72] O. Gafni, A. Polyak, O. Ashual, S. Sheynin, D. Parikh, and Y. Taigman. "Make-a-scene: Scene-based text-to-image generation with human priors". In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2022.

[73] R. Gal, Y. Alaluf, Y. Atzmon, O. Patashnik, A. H. Bermano, G. Chechik, and D. Cohen-Or. "An image is worth one word: Personalizing text-to-image generation using textual inversion". In: *arXiv.org* abs/2208.01618 (2022).

[74] A. Geiger. *Lecture: Deep Learning*. 2022.

[75] V. Goel, E. Peruzzo, Y. Jiang, D. Xu, N. Sebe, T. Darrell, Z. Wang, and H. Shi. "PAIR-Diffusion: Object-Level Image Editing with Structure-and-Appearance Paired Diffusion Models". In: *arXiv.org* (2023).

[76] L. Goetschalckx, A. Andonian, A. Oliva, and P. Isola. "GANalyze: Toward Visual Definitions of Cognitive Image Properties". In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. IEEE, 2019.

[77] S. Gong, M. Li, J. Feng, Z. Wu, and L. Kong. "Diffuseq: Sequence to sequence text generation with diffusion models". In: *arXiv.org* (2022).

[78] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[79] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative Adversarial Networks". In: *Advances in Neural Information Processing Systems (NIPS)*. 2014.

[80] Gradio. *Gradio: Quick and Easy UIs for Machine Learning Models*. Accessed: 2023-04-17. 2021. URL: https://gradio.app/.

[81] T. G. Grigg, D. Busbridge, J. Ramapuram, and R. Webb. *Do Self-Supervised and Supervised Methods Learn Similar Visual Representations?* 2021. arXiv.org: 2110.00528.

[82] T. Grigoryev, A. Voynov, and A. Babenko. "When, Why, and Which Pretrained GANs Are Useful?" In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2022.

[83] G. Gu, S. Chun, W. Kim, H. Jun, Y. Kang, and S. Yun. "CompoDiff: Versatile Composed Image Retrieval With Latent Diffusion". In: *arXiv.org* (2023).

[84] Gwern. *Making Anime Faces with StyleGAN*. 2020.

[85] H. Ham, T. J. Jun, and D. Kim. "Unbalanced gans: Pre-training the generator of generative adversarial network using variational autoencoder". In: *arXiv.org* 2002.02112 (2020).

[86] E. Härkönen, M. Aittala, T. Kynkäänniemi, S. Laine, T. Aila, and J. Lehtinen. “Disentangling random and cyclic effects in time-lapse sequences”. In: *ACM Trans. on Graphics* (2022).

[87] E. Härkönen, A. Hertzmann, J. Lehtinen, and S. Paris. “GANSpace: Discovering interpretable gan controls”. In: *Advances in Neural Information Processing Systems (NEURIPS)* (2020).

[88] K. He, X. Zhang, S. Ren, and J. Sun. “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2015.

[89] J. Hessel, A. Holtzman, M. Forbes, R. L. Bras, and Y. Choi. “CLIPScore: A reference-free evaluation metric for image captioning”. In: *Proc. EMNLP*. 2021.

[90] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. “GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium”. In: *Advances in Neural Information Processing Systems (NEURIPS)*. 2017.

[91] J. Ho, A. Jain, and P. Abbeel. “Denoising diffusion probabilistic models”. In: *Advances in Neural Information Processing Systems (NEURIPS)* (2020).

[92] J. Ho, C. Saharia, W. Chan, D. J. Fleet, M. Norouzi, and T. Salimans. “Cascaded Diffusion Models for High Fidelity Image Generation”. In: *J. Mach. Learn. Res.* (2022).

[93] J. Ho and T. Salimans. “Classifier-free diffusion guidance”. In: *CoRR* abs/2207.12598 (2022).

[94] S. Hochreiter and J. Schmidhuber. “Long short-term memory”. In: *Neural computation* (1997).

[95] E. Hoffer, I. Hubara, and D. Soudry. “Train longer, generalize better: closing the generalization gap in large batch training of neural networks”. In: *Advances in Neural Information Processing Systems (NEURIPS)*. 2017.

[96] J. Howard and S. Ruder. “Universal language model fine-tuning for text classification”. In: *Association for Computational Linguistics (ACL)*. 2018.

[97] X. Huang and S. Belongie. “Arbitrary style transfer in real-time with adaptive instance normalization”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[98] X. Huang, Y. Li, O. Poursaeed, J. Hopcroft, and S. Belongie. “Stacked generative adversarial networks”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[99] Y. Huang, Y. Cheng, A. Bapna, O. Firat, D. Chen, M. Chen, H. Lee, J. Ngiam, Q. V. Le, Y. Wu, et al. “Gpipe: Efficient training of giant neural networks using pipeline parallelism”. In: *Advances in Neural Information Processing Systems (NEURIPS)* (2019).

[100] D. A. Hudson and C. L. Zitnick. “Generative Adversarial Transformers”. In: *arXiv.org* 2103.01209 (2021).

[101] S. Huver, X. Mei, T. Deyer, and Z. Liu. *RadImageGAN*. https://github.com/lzl199704/RadImageGAN. 2023.

[102] S. Ioffe and C. Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *Proc. of the International Conf. on Machine learning (ICML)*. 2015.

[103] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. "Image-to-image translation with conditional adversarial networks". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[104] A. Jahanian, L. Chai, and P. Isola. "On the "steerability" of generative adversarial networks". In: *Proc. of the International Conf. on Learning Representations (ICLR)*. OpenReview.net, 2020.

[105] M. Javaid. "TikTok's Bold Glamour Filter Effect and the Role of Generative AI". In: *The Washington Post* (Mar. 2023). Accessed: 2023-04-17. URL: https://www.washingtonpost.com/technology/2023/03/08/tiktok-bold-glamour-filter-effect-generative-ai/.

[106] J. Johnson, A. Alahi, and L. Fei-Fei. "Perceptual Losses for Real-Time Style Transfer and Super-Resolution". In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2016.

[107] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. Lawrence Zitnick, and R. Girshick. "Clevr: A diagnostic dataset for compositional language and elementary visual reasoning". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[108] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, et al. "In-datacenter performance analysis of a tensor processing unit". In: *Proceedings of the 44th annual international symposium on computer architecture*. 2017.

[109] O. Kafri, O. Patashnik, Y. Alaluf, and D. Cohen-Or. "StyleFusion: A Generative Model for Disentangling Spatial Segments". In: *ACM Trans. on Graphics* (2022).

[110] A. Karnewar and O. Wang. "Msg-gan: Multi-scale gradients for generative adversarial networks". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[111] T. Karras, T. Aila, S. Laine, and J. Lehtinen. "Progressive growing of GANs for improved quality, stability, and variation". In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2018.

[112] T. Karras, M. Aittala, T. Aila, and S. Laine. "Elucidating the Design Space of Diffusion-Based Generative Models". In: *Advances in Neural Information Processing Systems (NEURIPS)*. 2022.

[113] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila. "Training Generative Adversarial Networks with Limited Data". In: *Advances in Neural Information Processing Systems (NEURIPS)*. 2020.

[114] T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen, and T. Aila. "Alias-Free Generative Adversarial Networks". In: *Advances in Neural Information Processing Systems (NEURIPS)*. 2021.

[115] T. Karras, S. Laine, and T. Aila. "A style-based generator architecture for generative adversarial networks". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[116] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. "Analyzing and Improving the Image Quality of StyleGAN". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. Computer Vision Foundation / IEEE, 2020.

[117] D. P. Kingma and M. Welling. "Auto-encoding variational bayes". In: *arXiv.org* (2013).

[118] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2015.

[119] D. P. Kingma and P. Dhariwal. "Glow: Generative Flow with Invertible 1x1 Convolutions". In: *Advances in Neural Information Processing Systems (NEURIPS)*. Ed. by S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. 2018.

[120] U. Kocasari, A. Dirik, M. Tiftikci, and P. Yanardag. "StyleMC: Multi-Channel Based Fast Text-Guided Image Generation and Manipulation". In: *Proc. of the IEEE Winter Conference on Applications of Computer Vision (WACV)* (2022).

[121] A. Kolesnikov, L. Beyer, X. Zhai, J. Puigcerver, J. Yung, S. Gelly, and N. Houlsby. "Big transfer (bit): General visual representation learning". In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2020.

[122] J. Kong, J. Kim, and J. Bae. "Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis". In: *Advances in Neural Information Processing Systems* 33 (2020).

[123] S. Kornblith, J. Shlens, and Q. V. Le. "Do better imagenet models transfer better?" In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[124] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, et al. "Visual genome: Connecting language and vision using crowdsourced dense image annotations". In: *International Journal of Computer Vision (IJCV)* (2017).

[125] A. Krizhevsky, G. Hinton, et al. "Learning multiple layers of features from tiny images". In: (2009).

[126] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks". In: *Communications of the ACM* (2017).

[127] K. Kurach, M. Lucic, X. Zhai, M. Michalski, and S. Gelly. "The gan landscape: Losses, architectures, regularization, and normalization". In: *Proc. of the International Conf. on Machine learning (ICML) Workshops*. 2018.

[128] T. Kynkäänniemi, T. Karras, M. Aittala, T. Aila, and J. Lehtinen. "The Role of ImageNet Classes in Fréchet Inception Distance". In: *CoRR* abs/2203.06026 (2022).

[129] T. Kynkäänniemi, T. Karras, S. Laine, J. Lehtinen, and T. Aila. "Improved Precision and Recall Metric for Assessing Generative Models". In: *Advances in Neural Information Processing Systems (NEURIPS)*. 2019.

[130] Lambda Labs. *All You Need Is One GPU: Inference Benchmark For Stable Diffusion*. 2022. URL: https://lambdalabs.com/blog/inference-benchmark-stable-diffusion.

[131] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[132] K. Lee, H. Chang, L. Jiang, H. Zhang, Z. Tu, and C. Liu. "Vitgan: Training gans with vision transformers". In: *Proc. of the International Conf. on Learning Representations (ICLR)* (2021).

[133] A. Lewkowycz, A. Andreassen, D. Dohan, E. Dyer, H. Michalewski, V. Ramasesh, A. Slone, C. Anil, I. Schlag, T. Gutman-Solo, et al. "Solving quantitative reasoning problems with language models". In: *arXiv.org* (2022).

[134] D. Li, H. Ling, S. W. Kim, K. Kreis, A. Barriuso, S. Fidler, and A. Torralba. "BigDatasetGAN: Synthesizing ImageNet with Pixel-wise Annotations". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2022).

[135] X. Li, J. Thickstun, I. Gulrajani, P. S. Liang, and T. B. Hashimoto. "Diffusion-lm improves controllable text generation". In: *Advances in Neural Information Processing Systems* 35 (2022).

[136] J. Liang, J. Cao, G. Sun, K. Zhang, L. V. Gool, and R. Timofte. "SwinIR: Image Restoration Using Swin Transformer". In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV) Workshops*. IEEE, 2021.

[137] J. H. Lim and J. C. Ye. "Geometric Gan". In: *arXiv.org* (2017).

[138] C.-H. Lin, J. Gao, L. Tang, T. Takikawa, X. Zeng, X. Huang, K. Kreis, S. Fidler, M.-Y. Liu, and T.-Y. Lin. "Magic3D: High-Resolution Text-to-3D Content Creation". In: *ArXiv* abs/2211.10440 (2022).

[139] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. "Feature Pyramid Networks for Object Detection". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[140] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. "Microsoft COCO: Common objects in context". In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2014.

[141] H. Ling, K. Kreis, D. Li, S. W. Kim, A. Torralba, and S. Fidler. "EditGAN: High-Precision Semantic Image Editing". In: *Advances in Neural Information Processing Systems (NEURIPS)* (2021).

[142] B. Liu, Y. Zhu, K. Song, and A. Elgammal. “Towards Faster and Stabilized GAN Training for High-fidelity Few-shot Image Synthesis”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2021.

[143] L. Liu, Y. Ren, Z. Lin, and Z. Zhao. “Pseudo numerical methods for diffusion models on manifolds”. In: *Proc. of the International Conf. on Learning Representations (ICLR)* (2022).

[144] X. Liu, C. Gong, L. Wu, S. Zhang, H. Su, and Q. Liu. “FuseDream: Training-free text-to-image generation with improved clip+ gan space optimization”. In: *CoRR* abs/2112.01573 (2021).

[145] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie. “A convnet for the 2020s”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[146] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu. “DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps”. In: *Advances in Neural Information Processing Systems (NEURIPS)*. 2022.

[147] Y. Ma, Y. He, X. Cun, X. Wang, Y. Shan, X. Li, and Q. Chen. “Follow Your Pose: Pose-Guided Text-to-Video Generation using Pose-Free Videos”. In: *arXiv.org* (2023).

[148] M. Marchesi. “Megapixel Size Image Creation using Generative Adversarial Networks”. In: *arXiv.org* abs/1706.00082 (2017).

[149] C. Meng, R. Gao, D. P. Kingma, S. Ermon, J. Ho, and T. Salimans. “On Distillation of Guided Diffusion Models”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* abs/2210.03142 (2022).

[150] L. Mescheder, A. Geiger, and S. Nowozin. “Which Training Methods for GANs do actually Converge?” In: *Proc. of the International Conf. on Machine learning (ICML)*. 2018.

[151] L. Mescheder, A. Geiger, and S. Nowozin. “Which training methods for GANs do actually converge?” In: *International conference on machine learning*. PMLR. 2018.

[152] L. Mescheder, S. Nowozin, and A. Geiger. “The numerics of gans”. In: *Advances in Neural Information Processing Systems (NEURIPS)* (2017).

[153] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. “Spectral Normalization for Generative Adversarial Networks”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2018.

[154] T. Miyato and M. Koyama. “cGANs with Projection Discriminator”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. OpenReview.net, 2018.

[155] S. Mo, M. Cho, and J. Shin. “Freeze the Discriminator: a Simple Baseline for Fine-Tuning GANs”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2020.

[156] R. Mokady, A. Hertz, K. Aberman, Y. Pritch, and D. Cohen-Or. "Null-text Inversion for Editing Real Images using Guided Diffusion Models". In: *arXiv preprint arXiv:2211.09794* (2022).

[157] E. Molad, E. Horwitz, D. Valevski, A. R. Acha, Y. Matias, Y. Pritch, Y. Leviathan, and Y. Hoshen. "Dreamix: Video diffusion models are general video editors". In: *arXiv.org* (2023).

[158] G. Mordido, H. Yang, and C. Meinel. "Dropout-gan: Learning from a dynamic ensemble of discriminators". In: *arXiv.org* (2018).

[159] S. Morozov, A. Voynov, and A. Babenko. "On Self-Supervised Image Representations for GAN Evaluation". In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2021.

[160] M. F. Naeem, S. J. Oh, Y. Uh, Y. Choi, and J. Yoo. "Reliable Fidelity and Diversity Metrics for Generative Models". In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. Proceedings of Machine Learning Research. 2020.

[161] C. Nash, J. Menick, S. Dieleman, and P. W. Battaglia. "Generating images with sparse representations". In: *Proc. of the International Conf. on Machine learning (ICML)*. 2021.

[162] B. Neyshabur, S. Bhojanapalli, and A. Chakrabarti. "Stabilizing GAN training with multiple random projections". In: *arXiv.org* 1705.0783 (2017).

[163] A. Nguyen, J. Yosinski, Y. Bengio, A. Dosovitskiy, and J. Clune. "Plug & Play Generative Networks: Conditional Iterative Generation of Images in Latent Space". In: *arXiv.org* 1612.00005 (2016).

[164] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever, and M. Chen. "Glide: Towards photorealistic image generation and editing with text-guided diffusion models". In: *Proc. of the International Conf. on Machine learning (ICML)* (2021).

[165] A. Q. Nichol and P. Dhariwal. "Improved Denoising Diffusion Probabilistic Models". In: *Proc. of the International Conf. on Machine learning (ICML)*. Proceedings of Machine Learning Research. PMLR, 2021.

[166] M.-E. Nilsback and A. Zisserman. "Automated flower classification over a large number of classes". In: *Proc. Indian Conf. on Computer Vision, Graphics & Image Processing*. 2008.

[167] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. "Wavenet: A generative model for raw audio". In: *arXiv.org* (2016).

[168] OpenAI. *DALL·E API*. 2022. URL: https://openai.com/product/dall-e-2.

[169] *OpenAI Labs*. https://labs.openai.com/. Accessed: April 18, 2023.

[170] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu. “Semantic image synthesis with spatially-adaptive normalization”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[171] O. Patashnik, Z. Wu, E. Shechtman, D. Cohen-Or, and D. Lischinski. “StyleCLIP: Text-Driven Manipulation of StyleGAN Imagery”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2021.

[172] W. Peebles, J.-Y. Zhu, R. Zhang, A. Torralba, A. A. Efros, and E. Shechtman. “Gan-supervised dense visual alignment”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022.

[173] G. Perarnau, J. van de Weijer, B. C. Raducanu, and J. M. Álvarez. “Invertible Conditional GANs for image editing”. In: *arXiv.org* (2016).

[174] E. Perot, P. de Tournemire, D. Nitti, J. Masci, and A. Sironi. “Learning to Detect Objects with a 1 Megapixel Event Camera”. In: *Advances in Neural Information Processing Systems (NEURIPS)*. 2020.

[175] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. “Deep contextualized word representations”. In: *Conf. of the North American Chapter of the Association for Computational Linguistics (NAACL)*. 2018.

[176] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall. “Dreamfusion: Text-to-3d using 2d diffusion”. In: *arXiv.org* (2022).

[177] P. Pope, C. Zhu, A. Abdelkader, M. Goldblum, and T. Goldstein. “The Intrinsic Dimension of Images and Its Impact on Learning”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. OpenReview.net, 2021.

[178] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. “Learning transferable visual models from natural language supervision”. In: *Proc. of the International Conf. on Machine learning (ICML)*. 2021.

[179] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. “Learning transferable visual models from natural language supervision”. In: *arXiv.org* 2103.00020 (2021).

[180] A. Radford, L. Metz, and S. Chintala. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. Ed. by Y. Bengio and Y. LeCun. 2016.

[181] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. *Improving language understanding by generative pre-training*. 2018.

[182] M. Raghu, T. Unterthiner, S. Kornblith, C. Zhang, and A. Dosovitskiy. “Do Vision Transformers See Like Convolutional Neural Networks?” In: *Advances in Neural Information Processing Systems (NEURIPS)*. 2021.

[183] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. “Hierarchical text-conditional image generation with CLIP latents”. In: *CoRR* abs/2204.06125 (2022).

[184] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun. “Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer”. In: *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)* (2020).

[185] S. Reed, Z. Akata, S. Mohan, S. Tenka, B. Schiele, and H. Lee. “Learning What and Where to Draw”. In: *Advances in Neural Information Processing Systems (NEURIPS)*. 2016.

[186] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. “Generative Adversarial Text to Image Synthesis”. In: *Proc. of the International Conf. on Machine learning (ICML)*. 2016.

[187] E. Richardson, Y. Alaluf, O. Patashnik, Y. Nitzan, Y. Azar, S. Shapiro, and D. Cohen-Or. “Encoding in style: a stylegan encoder for image-to-image translation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021.

[188] S. R. Richter, H. A. AlHaija, and V. Koltun. “Enhancing photorealism enhancement”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

[189] E. Robb, W.-S. Chu, A. Kumar, and J.-B. Huang. “Few-Shot Adaptation of Generative Adversarial Networks”. In: *arXiv.org* 2010.11943 (2020).

[190] D. Roich, R. Mokady, A. H. Bermano, and D. Cohen-Or. “Pivotal Tuning for Latent-based Editing of Real Images”. In: *arXiv.org* (2021).

[191] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. “High-resolution image synthesis with latent diffusion models”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[192] O. Ronneberger, P. Fischer, and T. Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. 2015.

[193] N. Ruiz, Y. Li, V. Jampani, Y. Pritch, M. Rubinstein, and K. Aberman. “Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation”. In: *arXiv.org* abs/2208.12242 (2022).

[194] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, S. K. S. Ghasemipour, B. K. Ayan, S. S. Mahdavi, R. G. Lopes, et al. “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”. In: *Advances in Neural Information Processing Systems (NEURIPS)*. 2022.

[195] C. Saharia, J. Ho, W. Chan, T. Salimans, D. J. Fleet, and M. Norouzi. *Image Super-Resolution via Iterative Refinement*. 2021. arXiv.org: 2104.07636.

[196] M. S. Sajjadi, O. Bachem, M. Lucic, O. Bousquet, and S. Gelly. “Assessing generative models via precision and recall”. In: *Advances in Neural Information Processing Systems (NEURIPS)*. 2018.

[197] M. S. Sajjadi, B. Scholkopf, and M. Hirsch. “Enhancenet: Single image super-resolution through automated texture synthesis”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2017.

[198] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. “Improved Techniques for Training GANs”. In: *Advances in Neural Information Processing Systems (NEURIPS)*. 2016.

[199] T. Salimans and J. Ho. “Progressive distillation for fast sampling of diffusion models”. In: *CoRR* abs/2202.00512 (2022).

[200] C. N. d. Santos, Y. Mroueh, I. Padhi, and P. Dognin. “Learning implicit generative models by matching perceptual features”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2019.

[201] M. B. Sariyildiz, K. Alahari, D. Larlus, and Y. Kalantidis. “Fake it till you make it: Learning transferable representations from synthetic ImageNet clones”. In: *CVPR 2023–IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023.

[202] A. Sasha Luccioni, C. Akiki, M. Mitchell, and Y. Jernite. “Stable Bias: Analyzing Societal Representations in Diffusion Models”. In: *arXiv.org* (2023).

[203] A. Sauer, K. Chitta, J. Müller, and A. Geiger. “Projected GANs Converge Faster”. In: *Advances in Neural Information Processing Systems (NEURIPS)*. 2021.

[204] A. Sauer and A. Geiger. “Counterfactual Generative Networks”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2021.

[205] A. Sauer, T. Karras, S. Laine, A. Geiger, and T. Aila. “StyleGAN-T: Unlocking the power of gans for fast large-scale text-to-image synthesis”. In: *Proc. of the International Conf. on Machine learning (ICML)* (2023).

[206] A. Sauer, K. Schwarz, and A. Geiger. “StyleGAN-XL: Scaling StyleGAN to large diverse datasets”. In: *Proc. SIGGRAPH*. 2022.

[207] E. Schonfeld, B. Schiele, and A. Khoreva. “A u-net based discriminator for generative adversarial networks”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[208] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, et al. “LAION-5B: An open large-scale dataset for training next generation image-text models”. In: *Advances in Neural Information Processing Systems (NEURIPS)*. 2022.

[209] K. Schwarz, Y. Liao, M. Niemeyer, and A. Geiger. “Graf: Generative radiance fields for 3d-aware image synthesis”. In: *Advances in Neural Information Processing Systems* 33 (2020).

[210] K. Schwarz, A. Sauer, M. Niemeyer, Y. Liao, and A. Geiger. "Voxgraf: Fast 3d-aware image synthesis with sparse voxel grids". In: *Advances in Neural Information Processing Systems* (2022).

[211] P. Sharma, N. Ding, S. Goodman, and R. Soricut. "Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning". In: *Proc. ACL*. 2018.

[212] Y. Shen, J. Gu, X. Tang, and B. Zhou. "Interpreting the Latent Space of GANs for Semantic Face Editing". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[213] Y. Shen, C. Yang, X. Tang, and B. Zhou. "InterFaceGAN: Interpreting the Disentangled Face Representation Learned by GANs". In: *IEEE Trans. Pattern Anal. Mach. Intell.* (2020).

[214] Y. Shen and B. Zhou. "Closed-Form Factorization of Latent Semantics in GANs". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[215] A. Shocher, Y. Gandelsman, I. Mosseri, M. Yarom, M. Irani, W. T. Freeman, and T. Dekel. "Semantic pyramid for image generation". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[216] Z. Si and S.-C. Zhu. "Learning hybrid image templates (HIT) by information projection". In: *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)* (2011).

[217] K. Simonyan and A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2015.

[218] A. Singh, R. Hu, V. Goswami, G. Couairon, W. Galuba, M. Rohrbach, and D. Kiela. "Flava: A foundational language and vision alignment model". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[219] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. "Deep unsupervised learning using nonequilibrium thermodynamics". In: *Proc. of the International Conf. on Machine learning (ICML)*. 2015.

[220] J. N. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli. "Deep Unsupervised Learning using Nonequilibrium Thermodynamics". In: *ArXiv* abs/1503.03585 (2015).

[221] J. Song, C. Meng, and S. Ermon. "Denoising Diffusion Implicit Models". In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2021.

[222] N. Spingarn, R. Banner, and T. Michaeli. "GAN "Steerability" without optimization". In: *Proc. of the International Conf. on Learning Representations (ICLR)*. OpenReview.net, 2021.

[223] D. Sungatullina, E. Zakharov, D. Ulyanov, and V. Lempitsky. "Image manipulation with perceptual discriminators". In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2018.

[224] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. “Going deeper with convolutions”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[225] M. Tan and Q. Le. “Efficientnet: Rethinking model scaling for convolutional neural networks”. In: *Proc. of the International Conf. on Machine learning (ICML)*. 2019.

[226] M. Tao, H. Tang, F. Wu, X.-Y. Jing, B.-K. Bao, and C. Xu. “DF-GAN: A Simple and Effective Baseline for Text-to-Image Synthesis”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[227] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li. “YFCC100M: The new data in multimedia research”. In: *Communications of the ACM* (2016).

[228] J. M. Tomczak. *Deep generative modeling*. Springer, 2022.

[229] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jegou. “Training data-efficient image transformers & distillation through attention”. In: *Proc. of the International Conf. on Machine learning (ICML)*. 2021.

[230] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou. “Training data-efficient image transformers & distillation through attention”. In: *Proc. of the International Conf. on Machine learning (ICML)*. Proceedings of Machine Learning Research. PMLR, 2021.

[231] H. Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, and H. Jégou. “Going deeper with image transformers”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2021.

[232] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. “Llama: Open and efficient foundation language models”. In: *arXiv.org* (2023).

[233] O. Tov, Y. Alaluf, Y. Nitzan, O. Patashnik, and D. Cohen-Or. “Designing an Encoder for StyleGAN Image Manipulation”. In: *ACM Trans. on Graphics* (2021).

[234] N.-T. Tran, V.-H. Tran, N.-B. Nguyen, T.-K. Nguyen, and N.-M. Cheung. “On data augmentation for GAN training”. In: *IEEE Trans. on Image Processing (TIP)* (2021).

[235] A. Van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al. “Conditional image generation with pixelcnn decoders”. In: *Advances in neural information processing systems* 29 (2016).

[236] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. “Attention is all you need”. In: *Advances in Neural Information Processing Systems (NEURIPS)* (2017).

[237] A. Voynov and A. Babenko. “Unsupervised Discovery of Interpretable Directions in the GAN Latent Space”. In: *Proc. of the International Conf. on Machine learning (ICML)*. PMLR, 2020.

[238] C. Wang, S. Chen, Y. Wu, Z. Zhang, L. Zhou, S. Liu, Z. Chen, Y. Liu, H. Wang, J. Li, et al. “Neural Codec Language Models are Zero-Shot Text to Speech Synthesizers”. In: *arXiv.org* (2023).

[239] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. “High-resolution image synthesis and semantic manipulation with conditional gans”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[240] X. Wang, L. Xie, C. Dong, and Y. Shan. “Realesrgan: Training real-world blind super-resolution with pure synthetic data supplementary material”. In: *Computer Vision Foundation open access* (2022).

[241] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. Change Loy. “Esrgan: Enhanced super-resolution generative adversarial networks”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2018.

[242] Y. Wang, C. Wu, L. Herranz, J. van de Weijer, A. Gonzalez-Garcia, and B. Raducanu. “Transferring gans: generating images from limited data”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. 2018.

[243] Z. Wang, H. Zheng, P. He, W. Chen, and M. Zhou. “Diffusion-gan: Training gans with diffusion”. In: *arXiv.org* (2022).

[244] Z. Wang, Q. She, and T. E. Ward. “Generative adversarial networks in computer vision: A survey and taxonomy”. In: *ACM Computing Surveys (CSUR)* (2021).

[245] W. Wu, Y. Zhao, M. Z. Shou, H. Zhou, and C. Shen. “DiffuMask: Synthesizing Images with Pixel-level Annotations for Semantic Segmentation Using Diffusion Models”. In: *arXiv.org* (2023).

[246] Y. Wu and K. He. “Group normalization”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[247] Z. Wu, D. Lischinski, and E. Shechtman. “StyleSpace Analysis: Disentangled Controls for StyleGAN Image Generation”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. Computer Vision Foundation / IEEE, 2021.

[248] K. Xiao, L. Engstrom, A. Ilyas, and A. Madry. “Noise or signal: The role of image backgrounds in object recognition”. In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2021.

[249] I. Z. Yalniz, H. Jégou, K. Chen, M. Paluri, and D. Mahajan. “Billion-scale semi-supervised learning for image classification”. In: *arXiv.org* (2019).

[250] S. Yang, L. Jiang, Z. Liu, and C. C. Loy. “VToonify: Controllable High-Resolution Portrait Video Style Transfer”. In: *ACM Transactions on Graphics (TOG)* 41.6 (2022).

[251] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao. “Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop”. In: *arXiv.org* 1506.03365 (2015).

[252] J. Yu, X. Li, J. Y. Koh, H. Zhang, R. Pang, J. Qin, A. Ku, Y. Xu, J. Baldridge, and Y. Wu. "Vector-quantized image modeling with improved VQGAN". In: *arXiv.org* (2021).

[253] J. Yu, Y. Xu, J. Y. Koh, T. Luong, G. Baid, Z. Wang, V. Vasudevan, A. Ku, Y. Yang, B. K. Ayan, et al. "Scaling autoregressive models for content-rich text-to-image generation". In: *CoRR* abs/2206.10789 (2022).

[254] N. Yu, G. Liu, A. Dundar, A. Tao, B. Catanzaro, L. Davis, and M. Fritz. "Dual Contrastive Loss and Attention for GANs". In: *arXiv.org* 2103.16748 (2021).

[255] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola. "Dive into deep learning". In: *arXiv.org* (2021).

[256] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena. "Self-attention generative adversarial networks". In: *Proc. of the International Conf. on Machine learning (ICML)*. 2019.

[257] H. Zhang, W. Yin, Y. Fang, L. Li, B. Duan, Z. Wu, Y. Sun, H. Tian, H. Wu, and H. Wang. "ERNIE-ViLG: Unified generative pre-training for bidirectional vision-language generation". In: *CoRR* abs/2112.15283 (2021).

[258] L. Zhang and M. Agrawala. "Adding conditional control to text-to-image diffusion models". In: *arXiv.org* (2023).

[259] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[260] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, et al. "Opt: Open pre-trained transformer language models". In: *arXiv.org* (2022).

[261] X. Zhang, S. Park, T. Beeler, D. Bradley, S. Tang, and O. Hilliges. "ETH-XGaze: A Large Scale Dataset for Gaze Estimation Under Extreme Head Pose and Gaze Variation". In: *Proc. of the European Conf. on Computer Vision (ECCV)*. Lecture Notes in Computer Science. Springer, 2020.

[262] J. Zhao, M. Mathieu, and Y. LeCun. "Energy-based generative adversarial network". In: *Proc. of the International Conf. on Learning Representations (ICLR)*. 2017.

[263] M. Zhao, Y. Cong, and L. Carin. "On leveraging pretrained GANs for generation with limited data". In: *Proc. of the International Conf. on Machine learning (ICML)*. 2020.

[264] S. Zhao, Z. Liu, J. Lin, J. Zhu, and S. Han. "Differentiable Augmentation for Data-Efficient GAN Training". In: *Advances in Neural Information Processing Systems (NEURIPS)*. 2020.

[265] S. Zhao, Z. Liu, J. Lin, J.-Y. Zhu, and S. Han. "Differentiable augmentation for data-efficient GAN training". In: *Advances in Neural Information Processing Systems (NEURIPS)*. 2020.

[266] Z. Zhao, Z. Zhang, T. Chen, S. Singh, and H. Zhang. “Image augmentations for GAN training”. In: *arXiv.org* 2006.02595 (2020).

[267] Y. Zhou, R. Zhang, C. Chen, C. Li, C. Tensmeyer, T. Yu, J. Gu, J. Xu, and T. Sun. “Towards Language-Free Training for Text-to-Image Generation”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[268] J. Zhu, Y. Shen, D. Zhao, and B. Zhou. “In-Domain GAN Inversion for Real Image Editing”. In: *Proc. of the European Conf. on Computer Vision (ECCV)*. Ed. by A. Vedaldi, H. Bischof, T. Brox, and J. Frahm. Springer, 2020.

[269] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. “Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks”. In: *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*. 2017.